

Segment - Technical Manual v1.8 R1172



July 16, 2010

MEDVISO AB
<http://www.medviso.com>
Kollegievägen 39
SE-224 73 Lund
Sweden
Tel: +46-76-183 6442

Contents

Contents	iii
1 Regulatory status	1
1.1 Commercial usage of Segment	1
1.1.1 Indications for use	1
1.2 Investigational purposes	2
2 License terms	3
3 Acknowledgements	5
4 Conventions and Abbreviations	7
4.1 Typographic conventions	7
4.2 Abbreviations	7
4.3 Trademarks	9
5 Document purpose	11
6 Coding Conventions	13
6.1 Naming functions	13
6.2 Naming variables	13
6.3 Name graphical handles	14
6.4 Object oriented programming	14
6.5 Graphical user interfaces	15
6.6 Indentation and spacing	15
6.7 Documenting code	15
7 Coordinate systems	17
7.1 Introduction	17

CONTENTS

8 Overview of Segment	19
9 Running Segment from Matlab	23
10 DATA variable	25
11 SET variable	35
12 How to create own plug-ins	57
13 Segment user community	59
14 Testing Segment	61
14.1 Testing functionality	61
14.2 Testing broken callbacks	61
15 Implementation details	63
15.1 Version handling	63
15.2 Numeric representations	63
15.3 Loading data and interpretation of DICOM tags	63
15.4 Segmentation algorithm	65
15.5 Volume calculations	65
15.6 Mass calculations	66
15.7 Calculation of BSA	66
15.8 Peak ejection/filling Rate	66
15.9 Wall thickness	66
15.10 Infarct size, extent and transmuralilty	67
16 Subfunctions in segment.m	69
17 Subfunctions in openfile.m	101
18 Subfunctions in export.m	107
19 Subfunctions in roi.m	109
20 Subfunctions in utility.m	113
21 Subfunctions in tools.m	115

22 General input output functions	121
22.1 myadjust.m	121
22.2 mybrowser.m	121
22.3 myclientserver.m	121
22.4 mycopyfile.m	122
22.5 mycountunique.m	122
22.6 mydel.m	122
22.7 mydir.m	123
22.8 mydisp.m	123
22.9 mydispexception.m	123
22.10 myexecutableext.m	123
22.11 myfailed.m	123
22.12 mygui.m	123
22.13 mymaximize.m	125
22.14 mymenu.m	125
22.15 mymkdir.m	125
22.16 mymovefile.m	126
22.17 mymsgbox.m	126
22.18 myoptimpartial.m	126
22.19 myrequirepc.m	126
22.20 myset.m	127
22.21 mystubfailed.m	127
22.22 mywaitbarclose.m	127
22.23 mywaitbarstart.m	127
22.24 mywaitbarupdate.m	127
22.25 mywarning.m	128
22.26 mywarningnoblock.m	128
22.27 myworkoff.m	128
22.28 myworkon.m	128
Bibliography	129

1 Regulatory status

Segment may be used for either investigational off label use or commercial purposes. Please see license terms which license form that apply to you. Users are also required to investigate the regulatory requirements pertinent to their country or location prior to using Segment. It is in the users responsibility to obey these statutes, rules and regulations.

1.1 Commercial usage of Segment

FDA approved versions of Segment are identified with a labelling upon start up displaying licence details and the FDA 510(k) number K090833. If your version does not display this information your version is not FDA approved and you need to contact Medviso AB to receive a such license.

Please note that there are features that are not included in the FDA approval. These functions are marked in the user manual with *'The functions described in this chapter is off label use and for investigational use only.'*

1.1.1 Indications for use

Segment is a software that analyzes DICOM-compliant cardiovascular images acquired from magnetic resonance (MR) scanners. Segment specifically analyzes the function of the heart and its major vessels using multi-slice, multi-frame and velocity encoded MR images. It provides clinically relevant and reproducible data for supporting the evaluation of the function of the chambers of the heart such as left and right ventricular volumes, ejection fractions, stroke volumes, peak ejection and filling rates, myocardial mass, regional wall thickness, fractional thickening and wall motion. It also provides quantitative data on blood flow and velocity in the arterial vessels and at the heart valves. Segment is tested on MR images acquired from both 1.5 T and 3.0 T MR scanners. The data produced by Segment is intended to be used to support qualified cardiologist, radiologist or other licensed professional healthcare practitioners for clinical decision making. **It is a support tool that provides relevant clinical data as a resource to the clinician and is not intended to be a source of medical advice or to**

determine or recommend a course of action or treatment for a patient.

1.2 Investigational purposes

None of the organizations/persons named in conjunction with the software can accept any product or other liability in connection with the use of this software for investigational purposes

2 License terms

For general license terms of the usage of the software, see the User Manual.

Though parts of Segment is be released in source code form, this does not imply that standard open source rules do apply. Segment is a commercial product and is provided free of charge to the research community as a service and without any associated rights. Medviso AB does not give you any rights to do commercial derivative works of it. Nor give it you the right to compile it to a distributable standalone form. See discussion on license terms in [1].

3 Acknowledgements

Even if this project started as a one man project, it has grown and it would never been possible without the help of many many people.

Financial support has been received from the Swedish Heart-Lung foundation, Swedish Research Council, local founds from Östergötland County, and Region of Scania.

I would like to acknowledge all the people that have put in feed back on usability and desired functionality, algorithm etc. Among others: Andreas Otto, Andreas Sigfridsson, Erik Bergvall, Erik Hedström, Henrik Haraldsson, Henrik Engblom, Håkan Arheden, Jan Engvall, Lars Wigström, Lisa Hård af Segerstad, Karin Markenroth Bloch, Marcus Carlsson, Martin Ugander, Mikael Kanski. Finally thank to you all Segment users in the research community that has inspired and contributed to the development.



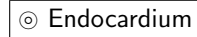
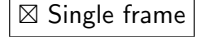
Special thanks to code providers Erik Bergvall (core routines of strain analysis), Helen Soneson (strain analysis module, SPECT module, Image fusion module), Shruti Agarwal (refactory of strain analysis module), Jonatan Wulcan (Sectra Plugin module and general improvements), Johannes Töger (3D flow and volume tracking), Mårten Larsson (3D flow and kinetic energy).

Commercial development has been done by Jane Sjögren (improvements to general object segmentation, implementation of prototype based segmentation, CT functionality, and graphical seriesselector). General debugging and implementation of the new interpolated contours has been done by Johan Ugander and Erik Södervall.

4 Conventions and Abbreviations

This chapter describes the typographic conventions and used abbreviations in this manual and in the program.

4.1 Typographic conventions

A	Key A at the keyboard.
Ctrl-A	Control key. Hold down Ctrl key and A simultaneously.
	Icon in toolbar.
*.tif	Filename extension.
C:/Program	Folder.
File	Menu, i.e File menu.
File→Save As	Sub menu, i.e under the File menu the item Save As is found.
	Push/Toggle button in the graphical user interface.
	Radiobutton in the graphical user interface.
	Checkbox in the graphical user interface.

4.2 Abbreviations

2CH	Two chamber view
3CH	Three chamber view
4CH	Four chamber view
3D	Three Dimensional
3D+T	Time Resolved Three Dimensional
AA	Ascending Aorta
ASW	Anterior Septal Wall Thickness
ARD	Aortic Root Diameter
BPM	Beats per minute
BSA	Body Surface Area
CO	Cardiac Output

CHAPTER 4. CONVENTIONS AND ABBREVIATIONS

CT	Computed Tomography
DA	Descending Aorta
DE-MRI	Delayed Enhancement MRI
ED	End diastole
EDD	End Diastolic Dimension
EDL	End Diastolic Length
EDV	End Diastolic Volume
EF	Ejection Fraction
ES	End systole
ESD	End Systolic Dimension
ESL	End Systolic Length
ESV	End Systolic Volume
FWHM	Full Width Half Maximum
GUI	Graphical User Interface
HR	Heart Rate
LV	Left Ventricle
LVM	Left Ventricle Mass
MO	Microvascular obstruction
MB	Mega Byte
MIP	Maximum Intensity Projection
MPR	Multiplanar Reconstruction
MR	Magnetic Resonance
MRI	Magnetic Resonance Imaging
PET	Photon Emission Tomography
PER	Peak Ejection Rate
PDW	Proton Density Weighted
PFR	Peak Filling Rate
PLW	Posterior Lateral Wall Thickness
ROI	Region Of Interest
RV	Right Ventricle
RVmaj	Right Ventricle Major Axis
RVmin	Right Ventricle Minor Axis
SPECT	Single Photon Emission Computed Tomography
SSFP	Steady State Free Precision
SV	Stroke volume
TOF	Time of Flight
VENC	Velocity Encoding limit

4.3 Trademarks

Below are some of the trademarks used in this user manual.

- Segment is a trademark of Medviso AB.
- Matlab is a trademark of the Mathworks Inc, (<http://www.mathworks.com>).

5 Document purpose

The purpose of this document is two fold; A) Work as a technical documentation of Segment for Medviso AB, B) work as a technical documentation for researcher who wish to use the plug-in functionality of Segmentas described in [1]. Therefore, all details described in this technical manual may not be applicable to all developers since parts of the source code is only accessible by Medviso AB.

6 Coding Conventions

This chapter describes the coding conventions that has been used when coding Segment. I have really tried to use a consistent naming rules to ensure that variables are given a logical name. However, there are occasions where historical reasons have lead to exceptions to the rules given below.

In general Segment originally used the global variable `NO` indicating the current image stack quite extensively. The use of `NO` is however not encouraged, and it is highly recommendable to in the function call indicate the current image stack. Work is ongoing to completely remove the global variable `NO`, but it takes time.

6.1 Naming functions

Functions are always named in lower case letters only. The underscore letter is not used with the exception of:

- Callbacks are named as `functionname_Callback`.
- Functions for mouse motion callbacks are given names ending with `_Motion`.
- Some functions have helper functions that are only called from other functions with the same name. These functions are named `_helper`.
- Functions for mouse button up/down callbacks are given names ending with `_Buttonup` or `_ButtonDown`.

Note that functions should not be ended with an `end` (Matlab supports both syntaxes). This is with the exception with objected oriented programming, see below.

6.2 Naming variables

The following rules apply:

- Global variables have all capital letters, i.e `DATA`.

- Field names of structures (and structure arrays) are given names starting with a capital letter. When two or more words are concatenated then the first letter of the second word also have a capital letter. For instance `DATA.CurrentTool`. Abbreviations are also given capital letters. For instance `SET(NO).TSize` where T stands for time.
- Field names with all capital letters corresponds to matrices. For instance `SET(NO).IM`, or `DATA.BALLOON`.
- The use of plural s is very limited and only used where there are limited numbers of values the field and it may be necessary to point out that there are more values than just a scalar. For instance `DATA.ViewPanels` is just instead of `DATA.ViewPanel` to point out there are (or may be) more than one view panel.

6.3 Name graphical handles

The graphical handles are named with all lower case letters and no underscore letters. Generally the names are rather long and exploratory. The following general naming conventions apply:

- Text edit boxes ends with `edit`
- Static text ends with `text`
- Axes objects ends with `axes`
- Pushbuttons ends with `pushbutton`
- Radiobuttons ends with `radiobutton`
- Checkboxes ends with `checkbox`
- Togglebuttons ends with `checkbox`

6.4 Object oriented programming

For new objected oriented functions the new syntax introduced in Matlab R2007b should be used. The old usage where the class definition is stored in a separate folder is obsoleted and existing code will be rewritten to new standard. Note that this new coding methods using `classdef` requires that an `end` statement is inserted in the end of the method code.

6.5 Graphical user interfaces

All new graphical user interfaces should be based on the class `mygui`. When doing so Segment automatically will remember the position of the user interface, and coding is considerably simplified so that one does not need to use persistent variables to get application data. Please consult the documentation of `mygui` for further details. Graphical user interfaces that do not use `mygui` should be rewritten to use `mygui`.

6.6 Indentation and spacing

Matlab standard indentation system is to be used, but with using the tab size to two spaces instead of the default four. Spacing and use of parenthesis is to be used to enhance readability. For instance

- `A = 12;` instead of `A=12;`.
- `if (a>b) || (d<e)` instead of `if a>b||d<e`

MLINT syntax guidelines should be followed and ideally when syntax hints are overridden, they should be motivated. For instance when MLINT reports; Warning data seems to grow inside a loop and that you should consider to preallocate for speed. Then before inserting a pragma to remove the warning you should make a note on why you did not preallocate (which usually is that the routine is not time critical, but is should generally be documented). Another example is when MLINT warns that the variable is unassigned, but it is assigned by loading a `.mat` file, then this should be commented.

The use of logical short cutting operands `||` and `&&` is strongly recommended.

6.7 Documenting code

The source code should be well documented so that new programmers easily can understand the code. Note, especially that the part of the code that was trickiest to write also deserves the most comments. Functions should be written as:

```
%-----  
%function out = foo(bar)
```

CHAPTER 6. CODING CONVENTIONS

```
%-----  
%Explanatory help text of the function.  
%Help text may span multiple lines.
```

Your code begins here

It is very important to follow this coding standard since part of this documentation is automatically generated from the source code.

7 Coordinate systems

This chapter specifies the coordinate systems used in Segment.

7.1 Introduction

DICOM specifies coordinates in the RL (right/left), AP (Anterior/Posterior) and FH (Feet/Head) coordinate system. Internally Segment uses a coordinate system (x,y,z) that is relative the respectively image stack. In other words for each image stack the relations between (RL,AP,FH) and (x,y,z) is different.

Throughout the program, the x dimension refers to the first dimension along the arrays (i.e the **rows** in Matlab). The upper left pixel in the images displayed on the screen have the coordinate (1,1). The unit used is in pixels. In the normal image coordinate system this means what is usually meant as the y coordinate if one think of coordinate systems learned in elementary school. For instance plotting commands therefore generally takes the form:

```
plot(SET(NO).EndoY(:,3,1),SET(NO).EndoX(:,3,1));
```

This will plot the contour of the endocardium of the third time frame and the first slice of the current image stack. The upper left most slice in the montage view is slice one. Furthermore it is assumed that the slices should be ordered so that going from the first slice to the last would be going from the base to the apex of the heart.

The relations between the (x,y,z) and (RL,AP,FH) coordinate systems is given by the fields `ImagePosition` and `ImageOrientation`. Below are two examples given.

The (RL,AP,FH) coordinates (center) of the voxel(s) `SET(1).IM(1,1,:,1)` are given by `SET(1).ImagePosition`.

The (RL,AP,FH) coordinates (center) of the voxel `SET(1).IM(2,3,4)` are

given by

```
zdir = cross(SET(1).ImageOrientation(1:3),SET(1).ImageOrientation(4:6))';  
pos = SET(1).ImagePosition(:)'+...  
      (2-1)*SET(1).ResolutionX*SET(1).ImageOrientation(4:6)'+...  
      (3-1)*SET(1).ResolutionY*SET(1).ImageOrientation(1:3)'+...  
      (4-1)*(SET(1).SliceThickness+SET(1).SliceGap)*zdir;
```

Coordinated conversions can be performed using the functions `xyz2rlapfh`,
and `rlapfh2xyz` in `Segment.m`.

8 Overview of Segment

Segment is written in Matlab, and is a fairly large software project. Currently it contains of around 100 000 lines of `m-code` distributed more than 100 files, and 1400 subfunctions, and 54 gui's. There are 23 files `c-code` with about 10 000 lines. This Technical Manual aims to describe how to write own plug-ins that interface with the software and give some implementation details for the interested reader. A good technical overview of the Segment project is also given in [1].

For copyright questions, see Chapter 2.

Before reading this user manual the reader should be well acquainted with Segment and programming in Matlab. This manual is **not** a manual how to use Segment, it's intention is only to give technical details on how things are implemented in Segment. Some implementational details are also given in the user manual.

The entire Segment block can be divided into 11 main function blocks. All these blocks are coded in a separate `m-file`. An overview on how they communicate and relates to each other is shown in Figure 1.

The main block is the 'Backbone' Segment. It contains the user interface, all graphical output routines, routines for volume calculations, etc. Generally this module calls the other functional blocks. These function blocks may then call helper functions such as graphical refresh or volume calculations etc in Segment, but they do not initiate new processes in the main block.

Some of the building blocks contains propriety algorithms and will only be made available as `p-code`, i.e platform independent precompiled Matlab code. The blocks that will be hidden are: `lv.m`, `rv.m`, `viability.m`, `flow.m`, `pacs.m`, `patientdatabase`. These functions will not be described in great detail in this document, for documentation of these files, please see internal documents at Medviso AB.

Segment uses three global variables to store the state of image data, graphi-

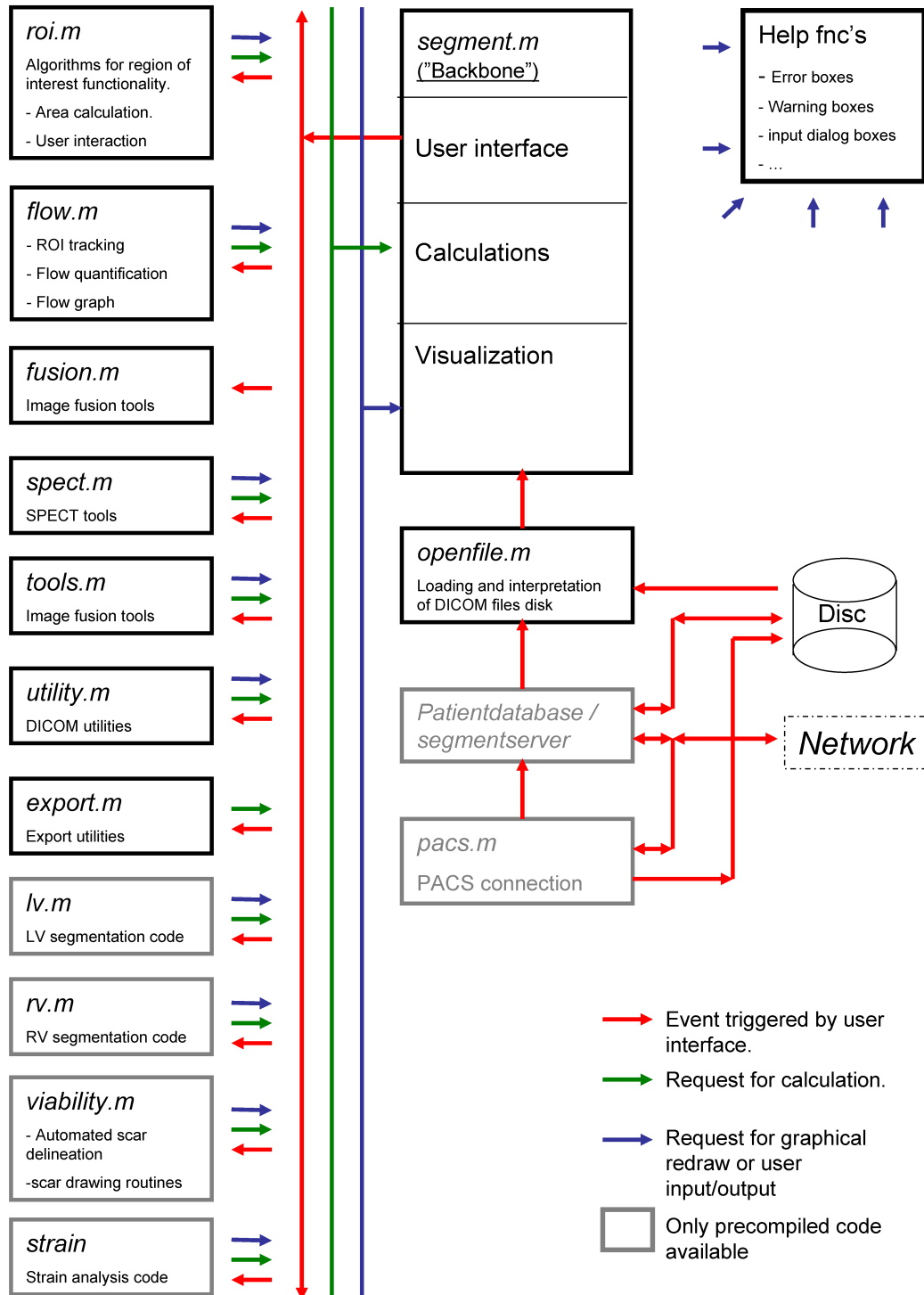


Figure 1: Overview of the main building blocks of Segment and transaction analysis. Red arrows indicate communication that is initiated from a user interface. Green arrow indicates call of calculation subroutines. Blue arrows indicate requests for graphical update or call of low level user input & output routines. Gray boxes indicate modules only available as object code.

cal handles. These three variable are **DATA**, **SET**, and **NO**. The variable **DATA** contains global state of the program and all main graphical handles. The variable **SET** contains all image data, segmentation, ROI's, measurements, annotation points, etc. The variable **NO** is simply a scalar and points to the current image stack. It is recommended not to use the global **NO**, but rather include this in the call. Rewriting this for larger parts of the code is a future enhancement of Segment.

The reason for having global variables instead of a completely functional programming approach is that:

- Simplicity for writing plug-ins.
- Execution speed. Using this approach there are way less variables/pointers that needs to be passed to the functions.

The drawback with using global variable is that the variable should at all times only reflect valid global states of the data. Therefore it is very important that modifications of the global variables are done with care.

9 Running Segment from Matlab

You need to have Matlab R2008a or later to run Segment. Running Segment from Matlab prompt is just as easy as running it as a stand-alone application. Note that necessary mex files have been compiled for Linux (both 64 and 32 bit), Mac OS X, and Windows (both 64 and 32 bit). When Matlab has started, simply type:

```
>> segment
```

To get access to the internal variable direct from the Matlab prompt, simply type:

```
>> global DATA SET NO
```


10 DATA variable

This global variable is used to store all the graphical handles, preferences, etc.

<code>ProgramVersion</code>	A string containing the version number. This string is initiated at startup.
<code>fig</code>	Handle to the main GUI figure.
<code>BlockingFigs</code>	A list of open figures that contains GUI's that are sensitive to changing the current image stack NO.
<code>imagefig</code>	Handle to the figure where the images are plotted. Currently this is same as <code>fig</code> , but are reserved for future use.
<code>DataLoaded</code>	True if image data is present.
<code>Silent</code>	True if no graphical output should occur. This is usefull when writing plugins that perform batch processes.
<code>Handles</code>	Struct with all graphical handles in the main GUI.
<code>InteractionLock</code>	When set to true, new calculation initiated from callbacks are prohibited. This may be obsoleted in the future and replaced with Matlab's queueing properties.
<code>Undo</code>	Struct containing undo information.
<code>UndoN</code>	Scalar that contains a pointer in the undo history list.

<code>LastSaved</code>	Time stamp when the image stack(s) last were saved. This is used for the auto-save functionality.
<code>Volrend</code>	Reserved for future use with the future volume rendering module.
<code>Colormap</code>	Current colormap used. This is a 256 x 3 array.
<code>ViewPanels</code>	A vector pointing to image stacks. If 2 x 2 panels are shown on the screen, then the length of <code>ViewPanels</code> is 4. For panels that does not contain any image stack the corresponding pointer is set to zero.
<code>ViewPanelsType</code>	<p>A cell array where each element contains the view mode for each image panel. Allowed view modes are:</p> <ul style="list-style-type: none"> • mode <code>one</code> shows the image stack as one slice. • mode <code>montage</code> shows all slices of the image stack arranged in a matrix style. • mode <code>mmodetemporal</code> shows the temporal part of a mmode image display. • mode <code>mmodespatial</code> shows the spatial part of a mmode image display. <p>The best way of changing viewing mode is to call the function <code>viewimage_Callback</code>.</p>
<code>ViewIM</code>	A cell array with the length same as <code>ViewPanels</code> . Each element contains a <code>uint8</code> array of size $N \times M \times T$ where T is the number of time frames, N and M are arbitrary numbers that depends on <code>ViewPanelsType</code> . This is used for graphical output and contains mapped intensities.

ViewMatrix	A two element vector with the size of the image panels (n x m).
CurrentPanel	Points to the current image panel. This number needs to be in the range $[1..N]$, where N is the number of valid panels.
CurrentTool	String containing the current tool. Valid tools are: <ul style="list-style-type: none">• point• measure• select• dragendo• dragepi• drawendo• drawepi• drawrvendo• drawrvepi• drawroi• drawscar• drawrubber• drawrubberpen• endopin• epipin• contrast• crop
NeedRedraw	True if the size of the main GUI have been changed.

Pref

Structure with preferences. For a complete description of the preferences, see the User Manual. The preferences are read when Segment is started or when the preferences GUI are opened. Segment tries to find a suitable place for the preferences file by looking in the environment variables APPDATA, USERPROFILE, HOMEPATH. Place can be found by calling the Segment function `getpreferencespath`. The fields in the **Pref** struct are:

- **datapath**. Path to image data. This is the path first opened when the fileloader GUI is started.
- **exportpath**. Path where image data is exported to.
- **AnonymMode**. True if patient details should no be shown on screen. Note that this does only affect the screen. To permanently anonymize an image stack, see User Manual for details.
- **AddPoints**. True if pins should be added when manually drawing a part of a contour.
- **EndoCenter**. True if the center of the endocardium is used for drawing spikes and regional wall motion analysis. If false the the center of the epicardial surface is used.
- **BlackWhite**. True if the lines contours should be drawn in white color instead of object specific colors.
- **LineWidth**. Width of the lines. Default is 1.
- **NumPoints**. Number of points to evaluate the endocontour along.
- **LearnMode**. True if learning messages should be displayed.
- **UndoHistory**. The maximum length of the undo history.

-
- **reportsheetpath.** Path to where the files for the report sheets are generated.
 - **IncludeAllPixelsInRoi.** If true, then also pixels that are touched by the ROI are included in the subsequent processing. Default is false, and in this mode only pixels whose centrum are inside the contour are included.
 - **AutoSave.** If true then the segmentation is autosaved every fifth minute under the name `autosave.seg`.
 - **ContourAdjustDistance.** The maximum distance to a contour a user can click before the contour is not acknowledged as a click on that contour. Measured in pixels.
 - **PacsTempStoragePath.** The path where the temporary files for the PACS retrieval are stored. This field might be obsoleted in future versions.

WindowSize	Size of the GUI in pixels.
Preview	Struct storing preview data. This struct is intensely used by the file open GUI. Generally the data is store in this struct and thereafter copied to the SET variable upon completion of the loading process. Therefore this struct also contains information on the last loaded/viewed image stack.
Run	True if a movie is playing.
Record	True if recording is active in the movie recorder GUI.

<code>LastPointer</code>	Type of the pointer displayed.
<code>ImagingTechniques</code>	Short names of the parameter files (*.par). This variable is created a startup. Therefore, Segment needs to be restarted before new parameter files can be used from the menu. However, the files themselves are not cached so the content in the files can be changed without restarting Segment.
<code>ImagingTechniquesFullNames</code>	Full names / titles of the parameter files. See above for details.
<code>ThisFrameOnly</code>	True if changes are applied only to the current timeframe. Example of functionality is segmentation, clearing segmentation etc. See details in the User Manual.
<code>ExcludePapillars</code>	True if the papillars should be excluded in the automatic segmentation. See User Manual and [2, 3] for details.
<code>UseLight</code>	True if to use current brightness and contrast as a cue in the Segmentation process. For further details see the User Manual.
<code>StartFrame</code>	First frame played in a movie. Initiated by <code>SET(NO).CurrentTimeFrame</code> , but needed for calculation what frame to display when playing a movie. See also <code>DATA.StartTime</code> .
<code>BalloonLevel</code>	Used to determine if the field <code>DATA.BALLOON</code> needs to be recalculated. Stores the parameter that was used last time to calculate <code>DATA.BALLOON</code> .
<code>BALLOON</code>	A matrix of the same size as <code>SET(NO).IM</code> containing the radial balloon force. For further details, see [2, 3].

<code>LevelSet</code>	Helper structure to <code>SET(NO).LevelSet</code> . <code>DATA.LevelSet</code> stores parameters that are pertinent to the graphical user interface that does not need to be stored in <code>SET(NO).LevelSet</code> .
<code>DATASETPREVIEW</code>	A $64 \times 64 \times N$ array containing the preview thumbnails, where N is the number of image stacks.
<code>EndoEDGE0</code>	Edge image of the endocardium in the first direction. For further details, see [2, 3].
<code>ENDOEDGE1</code>	See above.
<code>ENDOEDGE2</code>	See above.
<code>ENDOEDGE3</code>	See above.
<code>EpiEDGE0</code>	Edge image of the endocardium in the first direction. For further details, see [2, 3].
<code>EpiEDGE1</code>	See above.
<code>EpiEDGE2</code>	See above.
<code>EpiEDGE3</code>	See above.
<code>EndoEdgeDetected</code>	True if the edge images of the endocardium have been calculated.
<code>EndoEdgeDetected</code>	True if the edge images of the epicardium have been calculated.
<code>MovieFrame</code>	Temporary variable used to store one frame of a movie when recording a movie using the movie recorder in Segment.

NumPoints	Number of points along the endocardium, epicardium and ROI's. This default set to 80. This variable should be possible to change to get more points along a contour, but this have not been verified or tested. Note that changing this variable will most probably cause version incompatibilities with <code>.seg</code> and <code>.mat</code> files.
DataSetSelected	A logical vector with N elements where each element contains true for selected image stacks.
ImageTypes	<p>A cell array contains the different precode image types in Segment. They are:</p> <ul style="list-style-type: none"> • General • Perfusion Rest • Perfusion Stress • Strain FFE • Strain TFE • Late enhancement • Cine • Scout • Qflow • T2Stir • T1BB

The image types are used among others to find what image stack is what for automated batch processing of files. This list is subject to future changes.

ImageViewPlane	<p>A cell array contains the different precode image view planes in Segment. They are:</p> <ul style="list-style-type: none"> • 2CH • 3CH • 4CH • Sagittal • Coronal • Frontal • Transversal • Short-axis • RVOT • Aorta • Pulmonalis • Vena cava inferior • Unspecified <p>The image types are used among others to find what image stack is what for automated batch processing of files. This list is subject to future changes.</p>
BpInt	Intensity in the blood pool of the image stack SET(NO). This is used to avoid unnecessary recalculations of DATA.BALLOON.
MInt	Intensity of the myocardium of the current image stack. See above.
TimeStamp	Previously used for double click detection. Currently unused. May be removed in future versions.

11 SET variable

This global variable is probably the most important variable/structure since it contains all image data and all measurements.

The variable is a struct array. As an example, `SET(2)` refers to the second image stack. `SET(2).CurrentTimeFrame` refers to the field that contains the current time frame of the second image stack.

Below a list of all fields are given.

<code>IM</code>	This field contains the image data stored as a 4D <i>single</i> array. The order of the dimensions are $X \times Y \times T \times Z$. For more details on coordinate system conventions, see Section 7. The image data should lie between 0..1. For more details on image scaling see the fields <code>IntensityScaling</code> and <code>IntensityOffset</code> .
<code>XSize</code>	Size of image stack in X-direction. The following two expressions are equivalent <code>SET(NO).XSize</code> and <code>size(SET(NO).IM,1)</code> .
<code>YSize</code>	Size of image stack in Y-direction. The following two expressions are equivalent <code>SET(NO).YSize</code> and <code>size(SET(NO).IM,2)</code> .
<code>TSize</code>	Size of image stack in temporal direction. The following two expressions are equivalent <code>SET(NO).TSize</code> and <code>size(SET(NO).IM,3)</code> .
<code>ZSize</code>	Size of image stack in Z-direction (number of slices). The following two expressions are equivalent <code>SET(NO).ZSize</code> and <code>size(SET(NO).IM,4)</code> .

<code>StartSlice</code>	This field contains the first slice in the set of selected slices. For more details, see below.
<code>Endslice</code>	This field contains the last slice in the set of selected slices. The selected slices is then given by: <code>ind = SET(NO).StartSlice:SET(NO).EndSlice;</code> . The field may be an empty array when no slices are selected.
<code>CurrentTimeFrame</code>	This field contains the current time frame.
<code>CurrentSlice</code>	This field contains the current slice of the image volume.
<code>OrgXSize</code>	This refers to the original X-size of the images in the DICOM files. This is used to load <code>.seg</code> files to uncropped image data. The fields <code>OrgYSize</code> , <code>OrgTSize</code> , and <code>OrgZSize</code> have the same function in the other coordinate directions.
<code>rows</code>	This field contains the number of rows in the montage view of the current image stack.
<code>cols</code>	Same as above expect it contains number of columns instead.
<code>HeartRate</code>	Hear rate of the image stack. Note that different image stacks may have different heart rates. Unit is beats per minute. <code>HeartRate</code> is used to calculate cardiac output.
<code>BeatTime</code>	Controls how fast one heart beat is played when playing images as a movie. Originally set to <code>60/HeartRate</code> . May be adjusted with the slower/faster icons.

ResolutionX	Contains pixelspace in mm in X-direction. For discussion on coordinate systems, see Section 7. ResolutionY gives pixelsize in mm in Y-direction.
TIncr	Time increment in seconds between time frames. Must be non zero when image stack is time resolved.
TDelay	Trigger delay (i.e starting of image acquisition after trig pulse). Read from DICOM tags if presented otherwise set to zero. Given in seconds. Currently unused.
EchoTime	Echo time in MRI pulse sequence. Read from DICOM tags if presented otherwise set to zero. Given in milliseconds.
RepetitionTime	Repetition time in MRI pulse sequence. Read from DICOM tags if presented otherwise set to zero. Given in milliseconds.
InversionTime	Inversion time in MRI pulse sequence. Read from DICOM tags if presented otherwise set to zero. Given in milliseconds.
FlipAngle	Flip Angle in MRI pulse sequence. Read from DICOM tags if presented otherwise set to zero. Given in degrees.
NumberOfAverages	Number of averages acquired in MRI pulse sequence. Read from DICOM tags if presented otherwise set to zero.
VENC	Velocity encoding range in MRI pulse sequence. Read from DICOM tags if presented otherwise set to zero. Given in centimeters per second.
GEVENCSCALE	Special tag read from DICOM for GE scanners.

<code>SliceThickness</code>	Thickness of slices in millimeters. Read from DICOM file by looking at the tag <code>SliceThickness</code> . If not present, then the tags <code>ImageOrientation</code> and <code>ImagePosition</code> are used.
<code>SliceGap</code>	Gap between slices in millimeters (if present). Read from DICOM file by looking at the tag <code>SpacingBetweenSlices</code> if present. If not present, then the DICOM tags <code>ImageOrientation</code> and <code>ImagePosition</code> plus <code>SliceThickness</code> from above are used.
<code>Scanner</code>	Identified scanner identified by parsing the DICOM tag <code>Manufacturer</code> . Presented values are: <ul style="list-style-type: none">• ADAC• Bruker• GE• Philips• Siemens• Suinsa• Toshiba

ImagingTechnique String of capital letters identifying type acquisition used when acquiring the image stack. Possible values depends are given by the `.par` files. Each such files corresponds to one image type and contains information of how image should be mapped before segmentation. The two first letters should correspond to imaging modality (i.e MR, CT, PT, US, CR...). Shipped `.par` files are:

- `CTheart.par` (Segmentation of LV).
- `MRBB.par` (MR black-blood sequence).
- `MRDE.par` (MR delayed contrast enhancement).
- `MRGE.par` (MR gradient echo images).
- `MRPDW.par` (MR proton density weighted images).
- `MRSSFP.par` (MR steady state free precision, or fiesta).
- `MRSTIR.par` (MR STIR pulse sequence, edema sequence).
- `MRTOF.par` (MR Time of flight sequence for vessels).
- `NMBPSPECT.par` (Blood pool SPECT images).
- `OT.par` (Myocardial probability, obsoleted).
- `PT.par` (Generic for PET, not for segmentation use).
- `US.par` (Generic for ultrasound. Not for segmentation use).

ImageType	<p>Type of images that the image stack depicts. The field is used when identifying what image stacks to use for different analysis. If not set at loading then Segment tries to figure these details out by looking at what operations have been performed on what image stacks. Currently applied values are:</p> <ul style="list-style-type: none">• 'General' (Generic Image type if not specified/identified).• 'Perfusion Rest'• 'Perfusion Stress'• 'Strain FFE' (Strain Fast Field Echo).• 'Strain TFE' (Strain Turbo Field Echo).• 'Late enhancement' (Viability).• 'Cine'• 'Scout'• 'Qflow'• 'T2Stir'• 'T1BB'
-----------	--

<code>ImageViewPlane</code>	<p>View plane of images that the image stack depicts. The field is used when identifying what image stacks to use for different analysis. If not set at loading then Segment tries to figure these details out by looking at what operations have been performed on what image stacks. Currently applied values are:</p> <ul style="list-style-type: none"> • <code>'Unspecified'</code> (Generic Image view plane if not specified/identified). • <code>'2CH'</code> (Long axis 2 chamber view). • <code>'3CH'</code> (Long axis 3 chamber view). • <code>'4CH'</code> (Long axis 4 chamber view). • <code>'Sagittal'</code> • <code>'Coronal'</code> • <code>'Frontal'</code> • <code>'Transversal'</code> • <code>'Short-axis'</code> • <code>'RVOT'</code> • <code>'Aorta'</code> • <code>'Pulmonalis'</code> • <code>'Vena cava inferior'</code>
-----------------------------	--

<code>ImagePosition</code>	<p>Same as DICOM tag <code>ImagePosition</code>. If not presented then set to 0 0 0. Contains position in mm in 3D of the upper left pixel in the top slice in the image stack. For further details on coordinate systems see Section 7.</p>
----------------------------	--

ImageOrientation	Same as DICOM tag ImageOrientation. If not presented then set to 1 0 0 0 1 0. The three first numbers contains the normalized vector of Segments Y-direction given in the patient/table coordinate system. The last three numbers contains the direction of the X-direction. For further details on coordinate systems see Section 7.
Modality	Same as the corresponding DICOM tag.
PathName	Path to where the image is saved.
FileName	Filename of the file storing this image stack.
OrigFileName	Original filename. Note that this information is removed when anonymizing an image stack.

PatientInfo	<p>PatientInfo is a struct containing the following fields:</p> <ul style="list-style-type: none"> • Name is the name of the patient. • ID is the PatientID. • BirthDate in the date format 'YYYYMMDD'. • Sex. Applied values are '', 'M', 'F', '-'. • Age. Applied values are [], '', numeric, '78Y'. • HeartRate. Same as SET(NO).HeartRate, retained for backwards compability. • AcquisitionDate. Applied values, '', [], and 'YYYYMMDD'. • BSA. Applied values 0 or numeric value. Manually entered or automatically calculated from Weight and Length. See user manual for details and equations used. • Weight. Measured in kilograms. Applied values are 0 or numeric value. • Length. Measured in centimeters. Applied values are 0 or numeric value.
XMin	<p>When loading images it is possible to crop the images. XMin contains the number of pixels that the contours are translated due to cropping compared to the original image size. When no cropping is performed XMin is 1.</p>
YMin	<p>Same as XMin except relates to the Y-direction.</p>
Cyclic	<p>True for image stacks that are cyclic, i.e covers a complete heart cycle. This flag is used by the segmentation engine when delineating the left ventricle.</p>

Rotated	True for image stacks that are rotated around a common rotation axis. This is currently implemented as an add-on and only affects the volume calculation. Currently 3D visualization does not take this flag into account. This will be addressed in future versions.
---------	---

Scar

This is a struct containing information for viability analysis. For details on how the algorithm works, see [4]. The most important fields of the struct are:

- **IM** contains the image itself (a copy). This is from historical issues and may be removed in future versions.
- **Auto** a logical array that contains true in the positions that the algorithm marks as 'infarcted'.
- **Manual** an `int8` array that contains 1 for the pixels that the user have manually to be infarcted and -1 for the pixels that the user have marked as non-infarcted.
- **Result** a logical array containing the final viability delineation with the manual corrections.
- **NoReflow** a logical array containing one in the regions that have been identified as no reflow regions, or regions with microvascular obstruction.
- **MyocardMask** a logical array containing true in the myocardium and false otherwise.
- **beta** a scalar controlling the ruffness of the surface. See the algorithm description for further details [4].
- **stdlimit** a scalar determine the number of standard deviations from remote to take.
- **radius** Radius of the fast level set algorithm, see [4] for details.
- **Percentage** a scalar representing the percentage of the pixels that are marked as infarcted.
- **OnlyEndo** true if only endocardium delineation exists.

Flow

This struct contains information regarding how different image stacks are related to store flow information. Generally all information that are common for all the coupled image stacks are only stored in the magnitude image stack to avoid data redundancy. One example of this is storage of ROI's that are only performed in the magnitude image stack. The **Flow** struct contains the following fields:

- **Angio** contains a reference to which image stack that contains the angio imag. This angio image is essentially the absolute value of the velocity times the image magnitude and could be useful for vessel identification. Normally this field is set to an empty vector as the angio image is seldom constructed.
- **MagnitudeNo** contains reference to which image stack contains the magnitude information. For the image stack containing the magnitude information this points to itself.
- **PhaseNo** contains reference to the image stack containing the through plane flow information. Naming is somewhat inconsequent, by kept for legacy reasons. A better name would be **PhaseZ**.
- **PhaseX** contains reference to the image stack containing the flow in the X direction (Segment coordinate system).
- **PhaseY** contains reference to the image stack containing the flow in the Y direction (Segment coordinate system).

Then there are also a few fields that are optional and only available when eddy current compensation has been performed. These fields are:

- **PhaseCorr** contains the phase offset for the current direction (i.e if the current image stack is velocity/phase in the X direction, then **PhaseCorr** contains phase offset in the X direction, and so fourth. This field is always present, and if not applicable only an empty matrix is stored.
- **PhaseCorrPercentiles**, level of which percentiles to include in the detection of the static tissue. See documentation on the eddy current compensation for further details.
- **PhaseCorrMethod** contains the selected method for the eddy current compensation.
- **PhaseCorrTimeResolved** true if the eddy current compensation is time resolved. Default value is false. If time resolved, then **PhaseCorr** has the same image dimensions as **IM** otherwise the third image dimension is one.
- **PhaseCorrStaticTissueRois** true if regions where static tissue should be taken from drawn ROIs instead of estimated from the image. Default value is false.
- **VelMag** contains a reference to the image stack that contains a velocity magnitude stack (absolute value of the velocity). Normally this field is set to an empty vector as the velocity magnitude is seldom constructed.

Report	Field used to contain information for the patient report sheet generator functionality. This field is only set for the first image stack (i.e SET(1)). The exact content of this struct is subject to change.
Perfusion	Reserved for future usage.
Strain	Reserved for future usage.
Stress	Reserved for future usage.
KeptSlices	This vector contains the slices that are kept after a slice removal operation. Set by slice-removing operations, but otherwise unused. Kept for future use.
CenterX	Position of the center mark (+).
CenterY	Position of the center mark (+).

ROI	<p>a struct that contains information of store region of interest (ROI). It contains the following fields:</p> <ul style="list-style-type: none"> • X, X-coordinates of regions of interests. They are stored in an array as $N \times T$, where N is the number of points along the contour (to be precise it is actually <code>DATA.NumPoints</code>), T is the number of time frames. For non time resolved ROI's X can also be a vector of length N. • Y, Y-coordinates of regions of interest. For details about size, see above. • Z, Z-coordinates of the region of interest. • Sign, sign of the region of interest. This is used for flow quantifications and is stored as a vector with elements that are either -1 or 1. • RoiName, name of the region of interest. • RoiLineSpec, line specification for the region of interests. Stored as a string as Matlab compatible line specifications. Examples are 'b-' (blue line), 'y:' (yellow dotted line).
EndoX	<p>X-coordinates of the left ventricle endocardium. Applied values are either [], or an array with size $N \times T \times Z$, where N is the number of points along the contour (80, but technically <code>DATA.NumPoints</code>, T is the number of timeframes, and Z is the number of slices.</p>
EndoY	<p>Y-coordinates of the endocardium.</p>

EndoPinX	<i>X</i> -coordinates of endocardial pins. Stored as a cell-array of vectors of doubles with the size $T \times Z$, where T is the number of time frames, and Z is the number of slices. For more details about pins, see the User Manual.
EndoPinY	<i>Y</i> -coordinates of epicardial pins.
EndoXView	A compact representation used for drawing the endocardial contour in montage view. Stored as a double array with the size $((N+1)*Z) \times T$, where N is the number of points along the endocardial contour (<code>DATA.NumPoints</code>), T is the number of timeframes, and Z is the number of slices. It is the same as <code>EndoX</code> but each slice offseted and packed separated with a <code>NaN</code> so that drawing of the endocardial contours can be done with a single command.
EndoYView	The correspondence for the <i>Y</i> -coordinates to <code>EndoXView</code> .
EndoPinXView	Compact representation for endocardial pins. Stored as a cell vector with the T elements containing vectors.
EndoPinYView	See above.
EpiX	Epicardial contour of the left ventricle, otherwise same as <code>EndoX</code> .
EpiY	See above.
EpiXView	Corresponding to <code>EndoXView</code> .
EpiYView	Corresponding to <code>EpiXView</code> .
EpiPinX	Corresponding to <code>EndoPinX</code> .

EpiPinY	Corresponding to EndoPinY.
EpiPinXView	Corresponding to EndoPinXView.
EpiPinYView	Corresponding to EndoPinYView.
EndoDraged	A logical array containing a 1 where the left ventricle endocardial contour have been dragged.
EpiDraged	Same as above, but for the epicardial contour.
RVEndoX	Endocardial contour of the right ventricle. See also EndoX.
RVEndoY	See above.
RVEndoXView	Same as EndoXView, but for the right ventricle endocardium.
RVEndoYView	See above.
RVEpiX	Epicardial contour of the right ventricle. See also EpiX.
RVEpiY	See above.
RVEpiXView	Same as EpiXView, but for the right ventricle endocardium.
RVEpiYView	See above.
SectorRotation	Rotation of the Sector compartmentization of the left ventricle used for regional myocardial analysis. Unit is degrees. This is can be manually adjusted from the bulls-eye GUI.
Mmodex	X-coordinate for the center of the mmode line.

Mmodey	X -coordinate for the center of the mmode line.
Mmodelx	Direction coefficient of the mmode line.
Mmodely	Direction coefficient of the mmode line.
Mmodem1	Distance from the mmode line center to the first measurement point. Usually a positive value.
Mmodem2	Distance from the mmode line center to the second measurement point. Usually a negative value.
LVV	Volume inside the left ventricle represented as a $1 \times T$ vector of scalars, where T is the number of time frames.
EPV	Volume inside the epicardial volume of the left ventricle.
PV	Volume of the papillary muscles. See User Manual for further details how that is calculated.
LVM	The left ventricle myocardial volume is calculated as $EPV-LVV-PV$. Unit is ml.
ESV	Left ventricle end systolic volume.
EDV	Left ventricle end diastolic volume.
EDT	Left ventricle end diastolic time.
EST	Left ventricle end systolic time.
SV	Left ventricle stroke volume.
EF	Left ventricle ejection fraction.

PFR	Left ventricle peak filling rate. Calculated using circular convolution with a central difference (three elements) and taken as the largest derivative of LVV.
PER	Left ventricular peak ejection rate. Taken as the largest negative derivative of LVV.
PFRT	Time of peak filling. Given in time frames.
PERT	Time of peak ejection. Given in time frames.
RVV	Right ventricle volume. See LVV.
RVEPV	Right ventricle epicardial volume, see also EPV.
RVM	Right ventricle mass, see also LVM.
RVESV	Right ventricle end systolic volume, see also ESV.
RVEDV	Right ventricle end diastolic volume, see also EDV.
RVSV	Right ventricle stroke volume, see also SV.
RVEF	Right ventricle ejection fraction, see also EF.
Longaxis	Long-axis motion of the left ventricle, see User Manual for details on how the volume is compensated for long-axis motion. To convert to mm subtract with 1.
AutoLongaxis	True if the amount of long-axis compensation should be automatically calculated. See User Manual for details.
StartAnalysis	Time frame for start of analysis, used in flow quantification. See User Manual for details. Default value is 1.

<code>EndAnalysis</code>	End time of analysis, used in flow quantification. See User Manual for details. Default value is same as <code>TSize</code> .
<code>EndoCenter</code>	True if center of endocardium is used as center of the left ventricle when doing regional wall motion analysis. If false then the center of the epicardial volume is used. See User Manual for further details.
<code>NormalZoomState</code>	Four element vector describing the current zoom state for the image stack in normal mode. This representation is potentially subject to future changes.
<code>MontageZoomState</code>	Four element vector describing the current zoom state for the image stack in montage view mode. This representation is potentially subject to future changes.
<code>Measure</code>	<p>Struct that contains measurements places in the image stack. If no measures exist, then <code>Measure</code> is an empty array. If measures exist then it is a structure array where each measurement has the following fields:</p> <ul style="list-style-type: none"> • <code>X</code> A 2 x 1 vector with X-coordinates. • <code>Y</code> A 2 x 1 vector with Y-coordinates. • <code>Z</code> scalar with the slice. • <code>Length</code> Length of the measure in mm. • <code>Name</code> String containing the name of the measure.
<code>LevelSet</code>	Struct that contains general object segmentation module.

Point

Struct that contains position of annotation points.
The struct has the following fields:

- **X** $1 \times N$ vector with X -coordinates, where N is the number of annotation points.
- **Y** $1 \times N$ vector with Y -coordinates.
- **T** $1 \times N$ vector with time frames (i.e. one for each point).
- **Z** $1 \times N$ vector with slice (i.e. one for each point).
- **Label** $1 \times N$ cell array with string with the labels.

Originally all points were non time resolved. Thereafter where time resolved points implemented by making T points from one point. Non time-resolved points are shown as bold text. Time-resolved points are only implicitly connected by their position and more importantly the label.

IntensityMapping A struct containing information how the voxel values of the image stack is shown on the screen. The struct has the following fields:

- **Brightness**
- **Contrast**
- **Compression** Field reserved for future use.

The fields **Brightness** and **Contrast** translates into intensity according to the equation:

$$I = c(x_i - 0.5) + b \quad (1)$$

where c is contrast, b is brightness, x_i is the input intensity, and I is the output remapped intensity. The intensity I is clipped to the range $[0..1]$.
 $\text{map} = \text{contrast} * x_i + \text{brightness}$; $\text{map} =$ The field **Compression** is reserved for future use and will be used to implement image mappings that are sigmoid shaped.

IntensityScaling The fields **IntensityScaling** and **IntensityOffset** are used to convert the image data **IM** to the true data as presented in the DICOM images. The true data is calculated as:

$$z = I\alpha + \beta \quad (2)$$

where z is the true image data, I is the data stored in the field **IM**, α is **IntensityScaling**, and **IntensityOffset**.

IntensityOffset See above.

12 How to create own plug-ins

The easiest method of how to learn to make own plug-ins is to study the example `plugin_template.m`. Writing own plug-ins is a great way of spreading your algorithms to users all over the world and also to contribute to the Segment project. For more details on how to contribute, or learn more about interacting with other users, please see Chapter 13.

A plugin file must have a name beginning with `plugin_*.m`. Note that the plugin may of course call other functions that can reside anywhere on the Matlab path. When Segment is started the current folder is scanned for functions with this pattern. Matching functions are called with the argument `getname` and a string with the name that should appear in Segment menu is expected.

Currently there are two other plugins that are shipped with the standard Segment edition:

- Plugin to load non DICOM images into Segment. This plugin can load for instance `.jpg`, `.bmp`, `.tif`, `.png` files into Segment. This plugin gives some elementary details on the internal data structure.
- Plugin to calibrate image resolution. This plugin gives some hints on using own GUI's and also some details about the internal data structure in Segment.

For further documentation of these two plug-ins, please see the Segment User Manual.

13 Segment user community

As a result of the growing interest in Segment and as a response of numerous requests Medviso AB has started to form a user community web place. This initiative will be enlarged significantly as the members of the community both requests more and also expands the community. It is worth noting that in the user survey spring 2010, out of 169 answers 147 answered that they would follow the user community, and 45 answered that they would follow it often.

A preliminary start page of the user community can be found on the following Facebook page <http://www.facebook.com/pages/Segment/119840021370285>.

It is the aim to be able to provide the following activities on the user community pages:

1. Participate in discussion forums. Currently forums for Developers discussion and tips and tricks, Feature requests, Segment and Mac.
2. Contribute and share own plug-ins. This feature is currently not available. If you have plug-ins that you want to share, please email them to support@medviso.com and we will manually upload the plug-in. Currently writing own plug-ins to Segment is documented in the Segment Technical Manual, and two plug-in examples are documented in Chapter ??.
3. FAQ sections. Currently we are gathering FAQ in our support program. All (or almost all) support request will be made available in a searchable data base. Exceptions on when support requests are not included when the user request so in conjunction with classified projects.

Staff from Medviso AB will follow the user community page closely and monitor any incoming questions or uprising discussions.

If you have any ideas or suggestions on how we should improve the user community, please send us an email to support@medviso.com.

14 Testing Segment

To ensure highest quality of Segment as possible, a complete testing system has been implemented. The automated testing system produces a comprehensive test report of bugs found in Segment.

14.1 Testing functionality

The test functionality is implemented in the file `maketest.m`. It outputs the details of the testing result to the file `testsummarydoc.tex` and `testresultdoc.tex` that both are used when the testreport is generated. For an exact list of the included tests, see the latest available version of the test report. For each test performed a detailed list of tested functions are produced.

14.2 Testing broken callbacks

Testing broken callbacks is tested by the function `validatecallbacks`. Called with one input argument it tests the `.fig` file in the input argument and prints result in the standard output. If called with no input arguments, then the entire Segment project is tested and a report is written to the file `callbacksdoc.tex` which is used to produce the test report. The test script assumes that the first argument in a callback string is a sub function in the function name in the callback string (if applicable). The script tests `uimenu`, `uicontextmenu`, and `uicontrol` objects that has a callback string defined. To parse out sub functions of a function the function `document.m` is used.

15 Implementation details

This chapter contains the implementation details given in the User Manual.

15.1 Version handling

A proper version handling is employed when developing Segment. A detailed version history of Segment is available on the homepage

<http://segment.heiberg.se/version.htm>. Upto version 1.7 this version control was manually, now SVN with Tortoise as a front-end is used. For more detailed version history, please see the revision log of Segment SVN.

15.2 Numeric representations

All numbers are stored and used internally as double precision floating points with the following exceptions:

- Images are stored as single floats (normalized) or as integers (uint8), and then as they are stored in the DICOM files. Most functions in Segment will automatically convert the data to floats.
- Edge detection results are stored as integers (16 bits, 'normalized')
- Character strings are stored in 8bit ASCII format
- Infarct maps are stored as int8 (manual interaction), and uint8 (result).
- General segmentation tool store objects as levelset function with an uint8 representation where the zero levelset resides at 128.

Internally the image stack is normalized upon loading by a global maximum intensity such that all values are $[0..1]$. Offset and scaling is also calculated so that the image stack can be reconverted back to original signal intensities.

15.3 Loading data and interpretation of DICOM tags

This section describes how Segment interprets DICOM information to calculate important parameters such as geometric properties of the images.

- Number of slices. This is calculated from the presence of different slices based on the DICOM tags **ImagePosition** and **ImageOrientation**.
- Number of timeframes. This is based on dividing the total number of images with the number of slices.
- Time increment in ms between each timeframe. This is based on the difference between the number of timeframes divided by largest and the smallest value of the DICOM tag **TriggerTime**. If the DICOM tag **TriggerTime** is not present then the DICOM tag **TR** is used as time increment. Note that this might depend on your k-space acquisition scheme so for scanners that do not report **TriggerTime** you really need to double check the estimated value of time increment.
- Heart rate. The heart rate is taken from the DICOM tag **HeartRate** if present. Note that many vendors (including Siemens) does not specify this. As a fall back Segment tries to calculate the heart rate assuming full R-R interval coverage by using of trigger time (i.e it does not work for prospective imaging series). For long image acquisitions where one image is taken approximately for each heart beat then the heart rate is taken as the time between start of image acquisition and end of image acquisition adjusted for the number of frames. Note that in many cases this heart rate calculation will fail. Heart rate can be adjusted under patient details. Note also that heart rate may vary between image stacks therefore do not press Apply for all when manually changing heart rate. Heart rate is not used in any calculation, instead time increment between image frames is used in all calculations.
- Slice thickness in mm. The slice thickness is taken from the DICOM tag **SliceThickness**. If this tag is not present then the information is taken from same DICOM tags as number of slices, and assuming slice gap to be 0.
- Gap between slices in mm. This is taken from the DICOM tag **SpacingBetweenSlices**.
- Pixelspacing in X-direction in mm (vertical direction in Segment). This is taken from the DICOM tag **PixelSpacing**.
- Pixelspacing in Y-direction in mm (horizontal direction in Segment). This is taken from the DICOM tag **PixelSpacing**.
- Velocity encoding (VENC) in cm/s. For non velocity encoded images

this should be 0. How this is interpreted involves proprietary information of different scanner vendor information.

- Rotated image stack. This should by default be false. If your image stack is rotated, then change this to true. Currently this parameter is not taken from information in the DICOM tags and the user needs to manually change this when loading rotated image stacks.
- Cyclic image. If the image stack is cyclic, i.e. covers the whole heart cycle this should be true (default). For prospectively gated image series this should be false. This affects mainly the automated segmentation algorithm. Currently this information is not read from the DICOM information.

15.4 Segmentation algorithm

The method is more completely documented in Paper IV in the PhD thesis by Einar Heiberg [2].

The segmentation algorithm used by the program is a truly three-dimensional and time resolved deformable model specially adopted for segmentation of the left ventricle. Blood pool signal intensity is estimated from a user selected center point (see manually edit segmentation result). From this manually selected point the contour is initialized as a small time-resolved tube, and the contour of the tube is expanded until it reaches edges or when the local image intensity significantly changes from the estimated blood pool signal intensity.

15.5 Volume calculations

The volume calculations are done by a summing the area in each slice. The main reason for not using a more advanced volume integration method is that no one else is using that and therefore it might be difficult to compare the results. Segmentation (i.e. delineation of endocardium and epicardium) is stored on a sub-pixel accuracy and subsequent calculations are on a sub-pixel basis. For viability the classification into viable or scar is done on a pixel-wise basis and there the volume calculations are done by summing the number of pixels.

For the rotated image stacks the volume is given by a integration method. The volume contribution of each outline is given by :

$$\delta V = \frac{\pi}{2 * Z} \int y(s)^2 \text{sign}(y(s)) \frac{dy}{ds} ds \quad (3)$$

where the curve is given on a parametric representation $(x(s), y(s))$, Z is the number of slices in the rotated image stack. No long-axis compensation is performed for the rotated image stacks.

15.6 Mass calculations

When converting volume to mass the density is assumed to be 1.05 g/ml. Note that this number differs in the literature between 1.04 to 1.05. Furthermore, note that these numbers are valid for healthy myocardium ex-vivo, what happens in for instance infarcted regions is not shown in the literature. Therefore usually it is better to report volume instead of mass.

15.7 Calculation of BSA

The formula used is based on Graham and George.

$$BSE = w^{0.51456} * h^{0.42246} * 0.02350 \quad (4)$$

where w is the body weight in kg, and h is heigh in cm.

15.8 Peak ejection/filling Rate

When calculating peak ejection and peak filling rate the volume curve is differentiated using forward difference approximation. For cyclic datasets cyclic convolution is used for the calculation.

15.9 Wall thickness

Currently wall thickness is defined as the thickness along a radial spike from the endocardial or the epicardial center (depending on setting in the preferences. In the future I plan to also include the modified center line method. Note that the centers are calculated for each timeframe separately.

Wall thickening is defined as the wall thickness in end-systole minus the wall thickness in end-diastole. Note that it is possible to manually or automatically select what timeframes that are diastole or systole respectively.

Fractional wall thickening is defined as:

$$WT_f = \frac{WT - WT_{ED}}{WT_{ED}} \quad (5)$$

Where WT_f is fractional wall thickness and WT is wall thickness and WT_{ED} is wall thickness in end-diastole. In the bulls eye plot then fractional wall thickening is showed in end-systole.

15.10 Infarct size, extent and transmurality

Calculations of infarct sizes etc are based on 'counting' pixels, i.e. each pixel has a binary classification. There are two methods for regional analysis available, one are based where the percentage of the pixels that are inside the sector. The other method is based on radial spikes from the center (endo- or epicardial depending on setting in the preferences). The line between endocardium and epicardium is resampled in 50 steps and the percentage of infarcted pixels are counted.

Infarct extent is defined as the projected infarcted area on the endocardial surface [5].

$$I_{ext} = \sum_i \frac{T_i R_i}{R_i} \quad (6)$$

where I_{ext} is the infarct extent, T_i is the transmurality of sector i and R_i is the mean endocardial radius of sector i .

16 Subfunctions in `segment.m`

This chapter describes the subfunctions of `segment.m`.

<code>addmainicon_helper(callback,tooltip,cdata,tag,separator)</code>	Helper fcn to add an icon.
<code>addtopanels(no,mode)</code>	Finds an open space, otherwise increases number of panels.
<code>addviewicon_helper(callback,tooltip,cdata,tag,separator)</code>	Helper fcn to add an icon.
<code>z=anyall(a)</code>	Equivalent to <code>z = any(a(:));</code> .
<code>[x,y,name]=askcontour(queststri)</code>	Show menu so that user can indicate what contour to use. Used by <code>levelset</code> to import contours, and by <code>export</code> function to export contours as ascii file.
<code>autocontrast_Callback</code>	Automatically calculates contrast settings.
<code>autolongaxis_Callback</code>	Callback for automated long axis detection mode.
<code>im=bullseye(m,ax,n)</code>	Calculate bullseye from matrix <code>m</code> . <code>Ax</code> is optional axis where the output image <code>im</code> should be displayed.
<code>[varargout]=bullseye2(m,ax,n,flipx)</code>	Generate bullesye data. Optional output argument is <code>im</code> .
<code>[varargout]=bullseyeaha(m,ax,n,valuetype)</code>	Calculate and/or plot AHA 17 segment model.
<code>buttonup_Callback</code>	General buttonupfunction.

<code>bsa=calcbsa(weight,height)</code>	Calculates BSA. Formula based on Gehan and George.
<code>calcdatasetpreview</code>	Calculate thumbnails.
<code>rad=calcendoradius(no)</code>	Calculates endocardial radius.
<code>rad=calcepiradius(no)</code>	Calculates epicardial radius.
<code>[varargout]=calclvvolume(docomp)</code>	Calculate LV volume. Docomp if to use longaxis motion, see below. Uses $\text{area} * (\text{thickness} + \text{slicedist}) / 2$.
<code>calclvvolumepolar</code>	Calculate volume of lv when rotated image stacks.
<code>res=calcmypocardvolume(numsectors,no)</code>	Calculate myocardvolume.
<code>[xofs,yofs]=calcoffset(z,type)</code>	Calculate offset required to plot coordinates in viewing mode specified by type.
<code>[cellx,celly]=calcoffsetcells()</code>	Same as calculateoffset, but returns cells, used in updatemodeldisplay.
<code>[x,y]=calcplaneintersections(NO,no)</code>	Calculate intersections between image planes.
<code>radvel=calcradialvelocity(no)</code>	Calculate radial velocity of the endocardium. Forward difference is used.

<code>[meanarea,area]=calcroiarea(no,roino)</code>	Calculates roi area (helper fcn).
<code>[varargout]=calcrowscols(no,z)</code>	Calculate a good montage setup, so maximum number of slices can be displayed on the screen.
<code>[varargout]=calcrvvolume</code>	Calculate RV volume.
<code>calcrvvolumepolar</code>	Calculate RV volume when rotated image stacks.
<code>[xout,yout]=calcsegmentationintersections(no,type)</code>	Calculates intersections between segmentation in image stacks.
<code>ticks=calcticks(num,res)</code>	Helper fcn to calculate length of ticks in volume-graph.
<code>z=calctrueimage(im,no)</code>	Calculate true image intensities (as before Segment internal normalization).
<code>calcvolume(docomp)</code>	Calculate volume of segmentation and updates.
<code>wallthickness=calcwallthickness(sectors,no)</code>	Calculate wallthickness.
<code>[varargout]=cell2clipboard(outdata,writetofile)</code>	Converts a cell to a string that is output to clipboard. When too many output arguments it is written to a Excel file instead. If more than 8000 cells are written then an .xls file is written instead. Note that this used active-X on Windows and requires Excel to be installed on the computer.
<code>center_Buttondown(panel)</code>	Called when center '+' is pressed down, sets motion and buttonup fcn.

CHAPTER 16. SUBFUNCTIONS IN *SEGMENT.M*

<code>center_Buttonup</code>	This function is called when buttonup occurs after dragging center point.
<code>center_Motion</code>	Motion function of the center point.
<code>changewheel_Callback(h,e)</code>	Scrollwheel with modifier. Tab are not included but you can if you like.
<code>checkconsistency(timeframes,slice)</code>	Check consistency, to prevent earlier manual segmentations that have problems with direction left/right.
<code>corrupted=checkcorrupteddataforautomaticsave(setstruct)</code>	This function checks if the data (SET) is corrupted due to corrupted loading when loading files which has been saved with older saveversion see ticket 502 in wush for more details on the bug.
<code>cleardatalevelset(onlyprototype)</code>	Clear the struct DATA.LevelSet properly called when switching image stacks.
<code>clickedpin_Callback(type)</code>	Called when user clicks on a pin.
<code>contextmenu</code>	Context menu button down / callback.
<code>contrast_Callback(arg,panel)</code>	Activated by contrast tool, different from reset-light_Callback (above).
<code>mask=createmask(outsize,y,x)</code>	Function to generate a mask from a polygon represented with the vectors x and y.

<code>ctrlc</code>	Function to handle ctrl-c keypress, which is disabled.
<code>[x,y,z]=cyl2cart(xin,yin,slice,numsllices,rotationcenter)</code>	Convert from cylindrical to cartesian coordinates.
<code>doputpin_Callback(type)</code>	Put pins. Called when clicked, puts an pin and refines.
<code>drawall(n,m)</code>	Draws all panels that should be visible or up to the number specified as n.
<code>drawallslices</code>	This fcn updates graphics in all visible image panels.
<code>drawimagemontage(panel,viashow)</code>	Main workhorse for creating montage view.
<code>drawimageno(no)</code>	Call drawimagepanel for the panels that contain no including flow panels. If no is not specified then the current image stack NO is used. This function is typically called when new objects have been created or modified.
<code>drawimageone(panel,viashow)</code>	Main workhorse for creating one view.
<code>drawimagepanel(panel)</code>	Draws selected panels, used upon loading or when when major changes have occurred such as change of slice, added measurement, points etc. This fcn is the true workhorse in graphics etc.
<code>drawintersections</code>	Draw intersections between image stacks.
<code>drawroiinpanel(panel)</code>	Draw ROI's in one slice mode.

<code>drawsliceno(no)</code>	Call <code>drawslicepanel</code> for the panels that contain no including flow.
<code>drawslicepanel(panel)</code>	Update the image of the current slice giving viewing parameters from Workhorse of slice drawers.
<code>drawthumbnails(calculatepreview,sliderupdated)</code>	Draw all thumbnails. <code>Calculatepreview</code> is a boolean indicating if thumbnails needs to be re-drawn.
<code>drawvolume</code>	Draw volume curve.
<code>z=econvn(im,f)</code>	Function to filter image, uses fastest available convolver.
<code>ok=enablecalculation</code>	Returns ok, if slices are selected. Sideeffect of this function is that it turns on interaction lock, stops movie. Calling this function will not allow user to click on a new function before <code>endoffcalculation</code> is called. Note that this mechanism needs to be pretty safe with try/catch clauses otherwise Segment may hang. <code>Viewrefresh</code> calls <code>endoffcalculation</code> .
<code>endo_Buttondown(panel)</code>	Button down function for manual draw of endocardium.
<code>endoffcalculation</code>	Turns off interaction lock, called after a calculation. See above.
<code>epi_Buttondown(panel)</code>	Button down function for manual draw of endocardium.
<code>esed_Buttondown(type)</code>	Buttondown function when dragging ES or ED markers.

<code>esed_Buttonup(type)</code>	Buttonup function when dragging ES or ED markers.
<code>esed_Motion(type)</code>	Motion function when dragging ES or ED markers in volume graph.
<code>evalcommand_Callback</code>	Function to evaluate matlab commands from compiled version.
<code>y=existendo(no)</code>	True if endocardium exist in someslices.
<code>y=existendoinselected(no)</code>	True if endocardium exists in selected slices.
<code>y=existendoonlyinedes(no)</code>	True if endocardium exists in selected slices.
<code>y=existepiinselected(no)</code>	True if epicardium exist in selected slices.
<code>y=existepionlyinedes(no)</code>	True if endocardium exists in selected slices.
<code>y=existrvendoinselected(no)</code>	True if RV exist in selected slices.
<code>y=existrvendoonlyinedes(no)</code>	True if endocardium exists in selected slices.
<code>y=existrvepionlyinedes(no)</code>	True if endocardium exists in selected slices.
<code>fasterframerate_Callback</code>	Makes movie play faster.
<code>figure_DeleteFcn</code>	Shut down Segment in a controlled manner, and remove global DATA SET NO. Note that filequit callback takes care of asking user yesno.
<code>filecloseall_Callback(silent)</code>	

	Deletes all image stacks and resets Segment to original state.
<code>fileclosecurrentimagestack_Callback(frompreviewmenu)</code>	Close current image stack, i.e the current image stack is deleted. It also takes care of eventual cross couplings between image stacks.
<code>fileimportsegmentation_Callback</code>	Imports segmentation from a .seg file to current image stack. The difference is that less error checking is preformed. It also tries so find the matching slices etc.
<code>fileloadnext_Callback</code>	Load next .mat file in the current data folder.
<code>fileloadsegmentation_Callback(pathname,filename)</code>	Loads segmentation to current image stack from a .seg file.
<code>fileopen_Callback</code>	Loads a set of files as a 4D volume, and also updates preferences. Calls the fcn <code>openfile</code> which displays the fileloader GUI.
<code>filequit_Callback</code>	Quit segment, also ask user if he/she is sure.
<code>filesaveall_Callback</code>	Saves all image stacks to one .mat file. Calls <code>filesaveallas_Callback</code> which is the workhorse when saving image stacks.
<code>fail=filesaveallas_Callback(pathname,filename,topatientdatabase)</code>	Save all image stacks to the file specified. It also stores current view and modes etc.
<code>filesavecurrent_Callback</code>	Save current image set to file. Note that this is the old Segment file format and this fcn may soon be depreciated.

<code>filesavesegdicom_Callback</code>	Save image stack as DICOM file.
<code>filesavesegmentation_Callback(pathname,filename)</code>	Saves segmentation as a .seg file. This way of saving contours is not recommended and may be depreceiated.
<code>filesavetopacs_Callback</code>	Send image stacks to PACS. This function should display a list of available PACS (.con files) and when user has selected store files on disk temporarily and then send the files to the PACS.
<code>filesavetopatientdatabase_Callback</code>	Callback to save image stacks to patientdatabase. Uses functions in patientdatabase.
<code>cineshortaxisno=findcineshortaxisno</code>	Finds only one cine short axis stack.
<code>[res,varargout]=findmeaninsector(type,inp,pos,numsectors,no)</code>	Find indicies of points along the contour that corresponds to which sector. Then also calculated mean values of the input inp.
<code>[varargout]=findmeaninsectorslice(type,numpoints,tf,slice,numsectors,no)</code>	Find indicies of points along the contour that corresponds to which sector. Used when analysing the myocardium of short axis slices. Operates on given slice.
<code>[cineno,scarno,flowno,strainno,varargout]=findno</code>	

Find matching image stacks output normalno (normal short axis stack or closest equivalent) scarno (viability short axis stacks) flowno (flow image stack(s)) [stressbaseline] baseline stress images [stresswork] either medium work or maximum work.

`scarno=findscarshortaxisno`

Finds only one scar shortaxis stack.

`ind=findslicewithendo(no)`

Find slices with endocard in anytimeframe.

`ind=findslicewithendoall(no)`

Find slices with endocard in all timeframes.

`ind=findslicewithepi(no)` Find slices with endocard in anytimeframe.

`ind=findslicewithepiall(no)`

Find slices with endocard in all timeframes.

`ind=findslicewithrvendo(no)`

Find slices with RV endocard in anytimeframe.

`ind=findslicewithrvepi(no)`

Find slices with RV epicard in anytimeframe.

`ind=findslicewithscarall(no)`

Find and returns slices with scar.

`flow3dmovie.Callback(arg,scaling,no,up,down,left,right)`

Displays 3D movie of velocity in a vessel.

`flowbar_Buttondown(type)` Button down function for flow bar in flow report GUI.

`flowbar_Buttonup(type)` Button up function for flow bar in flow report GUI.

<code>flowbar_Motion</code>	Motion function for flowbar.
<code>flowcreate_helper(angio)</code>	Helper function to create flow derived image stacks.
<code>flowcreateangio_Callback</code>	Create angio image.
<code>flowcreatevelmag_Callback</code>	Create velocity magnitude image.
<code>flowdelete_helper(angio)</code>	Helper function to delete angio images.
<code>flowdeleteangio_Callback</code>	Delete angio image.
<code>flowdeletevelmag_Callback</code>	Delete velocity magnitude image.
<code>flowqpqs_Callback</code>	Qp/Qs flow analysis (not yet implemented).
<code>flowsetvenc_Callback</code>	Set VENC for current image stack.
<code>[varargout]=flowvesselenhancement(no)</code>	Beta function to show vessels with time depending flow.
<code>[x,y,slice]=getclickedcoords</code>	Find coordinates where the user last clicked. x and y are given in internal coordinate system, i.e the functions determines slice in montage view.
<code>no=getclickedpreview</code>	Function which returns the clicked preview image.
<code>[intersections, maxintersect]=getendointersection(no)</code>	

CHAPTER 16. SUBFUNCTIONS IN *SEGMENT.M*

	Returns the intersection of the endocardial segmentation and the current slice and current time frame of SET(no).
<code>r=getfieldifcommon(SET, fname)</code>	Helper function to filesavedicom_Callback.
<code>t=getframenumbers</code>	Calculates what frame to show when playing a movie if storing a movie then show next frame otherwise user timer info.
<code>[pos,xdir,ydir,zdir]=getimageposandorientation(no,slice)</code>	Return image orientation for current slice or second input argument. If two output variables, then zdir is also outputed.
<code>value=getthisframeonly</code>	Function to check thisframe only mode or not.
<code>help_Callback</code>	Open Segment homepage in browser.
<code>helpabout_Callback</code>	Displays help information about Segment.
<code>helpbug_Callback</code>	User bug report function.
<code>helphotkeys_Callback</code>	Shows help of hot keys in Segment global DATA.
<code>helpimagepos(im,sector,ofs,konst)</code>	Helper function to display image.
<code>helpimageposneg(im,sector,konst)</code>	Helper function to display image.
<code>helpopenlogfiles_Callback</code>	Open list of log files in browser.
<code>helpopenthislogfile_Callback</code>	

	Open log file for this session in browser.
<code>helpsupport_Callback</code>	Open mail composer to submit email to support.
<code>helpsupportreq_Callback(call)</code>	Callback to open support request GUI.
<code>helptutorials_Callback</code>	Opens the webpage for showing tutorials on Segment.
<code>helpusermanual_Callback</code>	Open user manual.
<code>highlighttool(h)</code>	Helper function to change color of a tool to represent that the tool is selected.
<code>iconcallbackhelper(handle,callback)</code>	Helper function to update icons.
<code>handle=iconcdatahelper(handle,icondata,tiptext)</code>	Helper function to update icons.
<code>fig=initializesegment(programversion)</code>	Initialization of Segment GUI.
<code>initmaintoolbar</code>	Initialization of main toolbar.
<code>initmenu</code>	Initialize the main menu, for instance adds extra utilities, and plugins.
<code>initviewtoolbar</code>	Initialize view toolbar.
<code>interpaddpoint</code>	Add interp point.
<code>interpdeletepoint</code>	Delete interp point.
<code>interpdeletepointall</code>	Delete interp points for all slices timeframes.

`interpdeletepointthisslice`

Delete interp points this slice.

`interpdeletepointthisslicephase`

Delete interp points this slice and phase.

`interpdrawGuessPoints_Callback(type,panel)`

Guess points in current selection then moves to corresponding tool.

`interpdraw_Buttondown(panel,type,forcedraw)`

Button down function to draw interp points.

`[interp_x,interp_y]=interpdrawhelper(interp_x,interp_y,contour_x,contour_y,x,y)`

Helper fcn to `interpdrawbuttonup` Side effects is update of `DATA.Pin`.

`[xout,yout]=interphelper(pinx,piny)`

Interpolates points to create a contour without loops.

`interppointButtonup(type)`

Button up function for interp points.

`interppointMotion(type)` Motion function for interp points.

`tool=interp toolfromcoords(x,y,slice)`

Get tool from point valid for 'interpendo' 'interpypi' 'interpvrando' 'interpvrapi'.

`keypressed(fignum,evnt)` Called when key is pressed.

`loadfieldhelper`

This fcn fixes backward compability when a .mat file is loaded.

`loadgui positions`

Load prestored positions of GUI's.

<code>loadok</code>	This function is called when managed to load an image volume It is essential for enabling, and setting up things.
<code>loadok_helper(no)</code>	Helper function to loadok* it enables vital GUI options.
<code>loadokmultidataset(no)</code>	This function is called when managed to load a multidataset.
<code>mainresize_Callback</code>	This fcn is called when user resizes GUI.
<code>makeviewim(panel)</code>	Rearrange data to show all slices.
<code>manualdraw_Buttndown(panel,type,new)</code>	Button down function for manual drawings.
<code>manualdraw_Buttonup(type,new,obj)</code>	Button up function for manual drawing.
<code>manualdraw_Motion</code>	Motion function for manual drawings.
<code>mccheckconsistency</code>	Check consistency, to prevent earlier manual segmentations that have problems with direction left/right.
<code>measure_Buttndown(panel)</code>	Button down function for measurements.
<code>measure_Buttonup</code>	Button up function for measurements.
<code>measure_Motion</code>	Motion function for measurements.
<code>stri=measureasklabel</code>	Asks for a label of a measurement.
<code>measureclearall_Callback</code>	Clear all measurements.

CHAPTER 16. SUBFUNCTIONS IN *SEGMENT.M*

<code>measureclearthis_Callback</code>	Clear current measurement.
<code>measureexport_Callback</code>	Export measurements.
<code>measuremove_Callback(dx,dy)</code>	Helper function to move measurements.
<code>measurepoint_Buttondown(panel)</code>	Buttondown function for a measurement point/marker.
<code>measurerenamethis_Callback</code>	Rename current measurement.
<code>mmodel1_Buttondown(panel)</code>	Called when mmodel1 is pressed down, sets motion and buttonup function.
<code>mmodel1_Motion</code>	Motion function of the first mmode point.
<code>mmodel1line_Buttondown(panel)</code>	Called when mmodel1line is pressed down, sets motion and buttonup function.
<code>mmodel1line_Motion</code>	Motion function of the first mmode line.
<code>mmodel2_Buttondown(panel)</code>	Called when mmodel1 is pressed down, sets motion and buttonup function.
<code>mmodel2_Motion</code>	Motion function of the second mmode point.
<code>mmodel2line_Buttondown(panel)</code>	Called when mmodel1line is pressed down, sets motion and buttonup function.
<code>mmodel2line_Motion</code>	Motion function of the second mmode line.

<code>mmode_Buttonup</code>	This function is called when buttonup occurs after dragging one of the mmode objects.
<code>mmodecenter_Buttondown(panel)</code>	Called when mmode center is pressed down, sets motion and buttonup function.
<code>mmodecenter_Motion</code>	Motion function of the mmode center point.
<code>mmodepoint1_Motion</code>	Motion function for mmodepoint one.
<code>mmodepoint2_Motion</code>	Motion function for mmodepoint two.
<code>mmodepoint1_Buttondown(panel)</code>	Called when mmodepoint1 is pressed down, sets motion and buttonup function.
<code>mmodepoint2_Buttondown(panel)</code>	Called when mmodepoint1 is pressed down, sets motion and buttonup function.
<code>mmodetimebar1_Buttondown(panel)</code>	Called when mmodepoint1 is pressed down, sets motion and buttonup function.
<code>mmodetimebar1_Motion</code>	Motion function for mmodepoint timebar.
<code>mmodetimebar2_Buttondown(panel)</code>	Called when mmodepoint1 is pressed down, sets motion and buttonup function.
<code>mmodetimebar2_Motion</code>	Motion function for mmodepoint timebar.
<code>montage_Buttondown(panel)</code>	Button down function for selecting slices in montage view.

CHAPTER 16. SUBFUNCTIONS IN *SEGMENT.M*

<code>montage_Buttonup(panel)</code>	Button up function for selecting slices in montage view.
<code>montage_Motion</code>	Motion function when selection slices in montage view.
<code>move_Buttonup(type,panel)</code>	Button down function for moving / translating objects / contours.
<code>move_Buttonup(type)</code>	Button up function for moving functions.
<code>move_Motion(reset)</code>	Motion function for moving / translating objects / contours.
<code>movealltowardsapex_Callback</code>	Changes CurrentSlice (and that of parallel image stacks) towards apex.
<code>movealltowardsbase_Callback</code>	Changes CurrentSlice (and that of parallel image stacks) towards base.
<code>movetowardsapex_Callback</code>	Change current slice towards apex.
<code>movetowardsbase_Callback</code>	Change current slice towards base.
<code>z=nanmean(a,dim)</code>	Same as mean in Matlab but handles also NaNs.
<code>res=nansum(a,dim)</code>	Same as sum, but ignores NaN's.
<code>nextallframe_Callback</code>	Displays next timeframe in current image panel and adjust all visible image stacks to the corresponding part of the cardiac cycle.

<code>nextframe_Callback</code>	Displays next timeframe of current image panel. Sideeffect is that the movie display is stopped if running.
<code>normal_Buttondown(panel)</code>	Called when button down in normal view.
<code>sameview=orientationcomparison(setindex1,setindex2)</code>	Compares the SET.ImageOrientation between two SETs. Help function to updateparallelsets.
<code>pan_Buttondown</code>	Button down function for panning of current image panel.
<code>pan_Buttonup</code>	Button up function for panning.
<code>pan_Motion(init)</code>	Motion function of pan.
<code>pin_Buttonup</code>	Button up function for pins.
<code>pin_Motion(type)</code>	Motion function for pins.
<code>[PinX,PinY,pinwarn]=pinresize(PinX,PinY,tf,slice,mean,f)</code>	Helper function to move pins when resizing contours.
<code>playall_Callback</code>	Starts movie display of all visible image stacks.
<code>playmovie_Callback</code>	Starts playing current image stack as a movie.
<code>plotmodelrot_Callback(daz,del)</code>	Changes view of 3D model of the segmentation.
<code>point_Buttondown(panel)</code>	Button down function when point tool is active.
<code>point_Buttonup</code>	Button up function for points.
<code>point_Motion</code>	Motion function for points.

<code>pointat_Buttondown(panel,type)</code>	Buttondown function when clicked at a point.
<code>pointclearall</code>	Clear all points.
<code>pointclearall_Callback</code>	Clear all points.
<code>pointcleartemplate_Callback</code>	Clear points using naming template.
<code>pointdeletethis_Callback</code>	Delete point.
<code>pointexportall_Callback</code>	Export all point data.
<code>ind=pointfind(silent)</code>	Find nearest point.
<code>pointfollow</code>	Follow a point over time.
<code>pointforward_Callback</code>	Track point forward in time.
<code>pointmaketimeresolvedthis_Callback</code>	Converts a none time resolved point to a time resolved point.
<code>pointmove_Callback(dx,dy)</code>	Helper function to move points.
<code>pointrenametemplate_Callback</code>	Rename points according to a renaming template.
<code>pointrenamethis_Callback</code>	Rename point.
<code>[x,y]=pointsfromcontour(X,Y,nbr_points)</code>	

	Select points from X,Y with probability higher when derivative magnitude in respect to distance to mass centre is large.
<code>pointshowthisframeonly_Callback</code>	Callback to make the current point visible in this time frame only.
<code>previousallframe_Callback</code>	Displays previous time frame in current image panel. For all other visible image stacks they are adjusted to show correspondig part of the cardiac cycle.
<code>previousframe_Callback</code>	Displays previous time frame of current panel.
<code>putpin_Buttondown(panel,type)</code>	Button down function for pins.
<code>putpin_Callback(panel,type)</code>	Callback to put pins.
<code>recursekeypressfcn(h,fcn)</code>	Helper function to create callbacks to keypressed function.
<code>z=remap(im,cmap,c,b)</code>	Remap data according to cmap.
<code>z=remapuint8(im,no,tempmap,c,b)</code>	

If SET(no).Colormap exists: Remap from image data to true color using color lookup. If not: Remap according to indexed grayscale Unless: An external colormap has been supplied, then use that. This is used to force truecolor grayscale, by passing true to RETURNMAPPING. c,b are contrast/brightness settings, used by contrast.Callback when doing realtime update during mouse motion.

report3dmodel_Callback(varargin)
GUI to show 3D display of segmentation.

report3dmodelrotated Display 3D model of rotated image stacks.

[varargout]=reportbullseye(fcn,arg2)
Main function for reporting bullseye.

[varargout]=reportflow_Callback(arg)
GUI for flow analysis.

reportradvel_Callback GUI for radial velocity.

reportroi_Callback(arg) GUI for ROI analysis.

reportslice_Callback(type)
GUI for graphical display of slice based regional function.

reportvolumecurve_Callback
GUI for reporting the volume curve.

reportvolumeloop_Callback
Reporter for volume loop (not yet implemented).

[xnew,ynew]=resampleclosedcurve(x,y,n)
General helper fcn to resample a closed curve.

<code>[xnew,ynew]=resamplemodel(x,y,n)</code>	Resample the a stack of contours (typically segmentation) to different number of points (n).
<code>resetguipositions</code>	Resets GUI positions of Segment.
<code>resetlight_Callback</code>	Activated by toolbar icon, different from contrast_Callback (below).
<code>resetpreview</code>	Reset preview structure in DATA.Preview.
<code>z=reshape2layout(im,no)</code>	Convert a 3D array to an layout:ed image with cols, and rows.
<code>map=returnmapping(no,forcetruecolor)</code>	If SET(no).Colormap exists: Returns truecolor colormap for this. If not: Returns grayscale 1-D indexcolor map. Unless: forcetruecolor is true, then 3-D truecolor grayscale is returned, regardless of SET(no).Colormap See REMAPUINT8 below.
<code>[pos]=rlapfh2xyz(no,rl,ap,fh)</code>	Convert from RL,AP,FH coordinates to Segment internal coordinate system.
<code>rotatedplothelper(xc,yc,linecolor)</code>	Helper function to display segmentation in rotated image stacks.
<code>saveguipositiontodisk</code>	Save guipositions to disk.
<code>scale_Buttondown(type,panel)</code>	Button down function for scaling of objects / contours.

CHAPTER 16. SUBFUNCTIONS IN *SEGMENT.M*

<code>scale_Buttonup(type)</code>	Button up function for scaling.
<code>scale_Motion(reset)</code>	Motion function for scaling.
<code>segmentclear_helper</code>	Helper fcn to clear segmentation.
<code>segmentclearall_Callback(force)</code>	Clear all segmentation, both endo and epi, lv and rv.
<code>segmentclearallbutsystoleanddiastole_Callback</code>	Clears all segmentation in all timeframes but systole and diastole.
<code>segmentclearalllv_Callback(force)</code>	Clear all lv segmentation, both endo and epi.
<code>segmentclearalllvbutsystoleanddiastole_Callback</code>	Clears all LV segmentation except in systole and diastole.
<code>segmentclearallrv_Callback(force)</code>	Clear all rv segmentation, both endo and epi.
<code>segmentclearallrvbutsystoleanddiastole_Callback</code>	Clears all RV segmentation except in systole and diastole.
<code>segmentclearslices(ind,timeframes,endo,epi,rvend,rvepi)</code>	Workhorse in clearing slices.
<code>segmentclearslices_Callback(endo,epi,rvend,rvepi)</code>	Helper function to clear segmentation in selected slices.
<code>segmentclearslicesthis_Callback(endo,epi,rvend,rvepi)</code>	Clear segmentation for slices in this timeframe.

`segmentclearthis_Callback(endo, epi, rvendo, rvepi)`

Helper function to clear segmentation in this (current slice) slice.

`segmentimportsegmentation_Callback(no)`

Import segmentation from another image stack. Imports to current image stack NO from no or if called with no input arguments user is asked.

`segmentimportsegmentationhelper(tono, fromno, doendo, doepe, dorvendo, dorvepi)`

Helper function to `segmentimportsegmentation_Callback`. - `tono` is destination of segmentation. - `fromno` is source.

`segmentremoveallinterp_Callback(silent, endo, epi, rvendo, rvepi, endoind, epiind, rvendoind)`

Remove all interp points.

`segmentremoveallpins_Callback(silent, endo, epi, rvendo, rvepi)`

Remove all pins.

`segmentremoveallpinsthisslice_Callback(current, endo, epi, rvendo, rvepi)`

Remove all pins in this slice.

`segmentremovepin_Callback(type, m)`

Removes pins.

`segmentremovethispins_Callback(endo, epi, rvendo, rvepi)`

Remove pins in this slice and timeframe.

`segmentreset edgedetection`

Reset the edge detection.

`segmentrvcopyfromlvendo_Callback`

Copy RV from LV segmentation (endocardium).

`segmentrvcopyfromlvepi_Callback`

Copy RV from LV segmentation (epicardium).

`segmentthisframeonly_Callback(silent)`

Sets segment in 'this frame only' mode. Changes made to segmentation, copying etc are performed only for the current timeframe.

`selectallslices_Callback` Selects all slices in current image stack.

`setcolormap_Callback(type)`

Set colormap for current image stack.

`setcurrenttimeframe(frame)`

Function to set current time frame, input argument is frame to be set to current time frame.

`setthisframeonly(value)` Callback to set this frame only mode.

`slowerframerate_Callback` Makes movie play slower.

`startlog(log)`

Start Segment log.

`stoplog`

Stop Segment logging.

`stopmovie_Callback`

End movie display.

`switchtoimagestack(no,force)`

This function makes no current image stack, and updates graphics.

`switchtopanel(panel)`

Make panel the currentpanel.

`switchtoslice(slice)`

Called when current slice has changed. If slice has changed, make sure montage/one are in sync Does nothing if slice = CurrentSlice.

<code>systolelistbox_Callback</code>	Function called when changing longaxis motion in list box.
<code>thumbnail_Butttdown</code>	Butttdown fcn for thumbnails.
<code>thumbnail_Buttonup(forceinfirstempty)</code>	Buttonup fcn for thumbnails.
<code>thumbnail_Motion</code>	Motion fcn for thumbnails.
<code>out=thumbnailno(in)</code>	Helper fcn to remember what image stack were clicked.
<code>thumbnailslider_Callback(whattodo)</code>	Slider callback for thumbnail slider.
<code>timebar_Butttdown</code>	Button down function for the time bar in the volume graph.
<code>timebar_Motion</code>	Update current timeframe when user clicked in volume graph.
<code>unlighttool(h)</code>	Helper function to change color of a tool to represent unselected.
<code>unselectallslices_Callback</code>	Unselects slices in current image stack.
<code>update_thumbnail(no)</code>	This fcn updates thumbnail no.
<code>updateicons(mode)</code>	Updates icons when new mode is selected.
<code>result=updateintersections_Callback(slices,no,type)</code>	Calculates the intersection of a endocardial segmentation and SET(no) for slices == 'all' or slices == 'current'.

<code>updatemmode(arg)</code>	Calculate and show mmode image.
<code>updatemmodeline</code>	Updates the mmode line in mmode display.
<code>updatemmodevisibility</code>	Updates mmode visibility.
<code>updatemodeldisplay(no)</code>	Update data used to correctly display segmentation in montage view. This fcn needs to be called when the segmentation has changed.
<code>updateoneim(no)</code>	Updates viewim for 'one view' or 'mmodespatial' Called when changed currentslice.
<code>updateparallelsets</code>	Chages CurrentSlice in SETs that are parallel to SET(NO) to the slice closest to SET(NO).CurrentSlice.
<code>updateselectedslices</code>	Graphically updates which slices are selected.
<code>updatetimethings(no)</code>	This fcn is called from switchimagestack and disables/enables features that are not available depending of the image stack is timeresolved or not.
<code>updatetitle</code>	Updates titleline of main GUI.
<code>updatetool(newtype,panel)</code>	Update display of tools and also callbacks of the graphical objects in the panel. If panel given then only update the panel, otherwise all image panels are updated.
<code>updateviewicons</code>	Enables/Disables icons depending on image data (nbr timeframes etc).
<code>updatevolume</code>	Calc volume of segmentation and graphically update.

<code>userlongaxis_Callback</code>	Callback to set user adjusted long axis motion.
<code>viewaddtoolbar_Callback</code>	Helper function to add a toolbar.
<code>viewallimagestacks_Callback</code>	Displays all image stacks.
<code>viewhideimagestack_Callback(no)</code>	Remove clicked image stack from view panels.
<code>viewhideinterp_Callback(varargin)</code>	Toggle visibility of countours from other image stacks.
<code>viewhideintersections_Callback(varargin)</code>	Toggle visibility of plane intersections.
<code>viewhidelv_Callback(varargin)</code>	Toggle visibility of lv segmentation.
<code>viewhidemeasures_Callback</code>	Toggle visibility of measurements.
<code>viewhideothercontour_Callback(varargin)</code>	Toggle visibility of countours from other image stacks.
<code>viewhidepanel_Callback</code>	Hides current panel.
<code>viewhidepap_Callback</code>	Toggle visibility of papillary overlay.
<code>viewhidepins_Callback(varargin)</code>	Toggle visibility of pins.
<code>viewhideplus_Callback(varargin)</code>	Toggle visibility of center +.

CHAPTER 16. SUBFUNCTIONS IN *SEGMENT.M*

<code>viewhideroi_Callback(varargin)</code>	Toggle visibility of roi's.
<code>viewhiderv_Callback(varargin)</code>	Toggle visibility of rv segmentation.
<code>viewhidescar_Callback(varargin)</code>	Toggle visibility of scar contours.
<code>viewhidetext_Callback(varargin)</code>	Toggle visibility of text.
<code>viewimage_Callback(type)</code>	Select type of image to view on screen.
<code>viewimagestack_Callback(no)</code>	Make clicked image stack visible.
<code>viewimagestackas_Callback(mode)</code>	Select how to view current image stack.
<code>viewintersections_Callback</code>	Toggle visibility of intersections.
<code>viewmanualinteraction_Callback</code>	Displays a GUI indicating in which slices and time-frames manual interaction have been made for LV segmentation.
<code>viewpandir_Callback(direction)</code>	Pans current view panel.
<code>viewrefresh_Callback</code>	Main graphical refresh.
<code>viewrefreshall_Callback</code>	Main refresh of GUI.
<code>viewspecial_Callback(mode)</code>	

	Called to switch to predefined layouts of the GUI. Mode is current mode to use.
<code>viewupdateannotext(panel)</code>	Updates the visibility of point/measurement text. Respects hideX settings.
<code>viewupdatetextposition(panel)</code>	Updates the location of the corner text, to always stay still. From <code>drawx()</code> , it is called panelwise, since the linked panels aren't ready when the first panel is drawn. From other situations, it is called without arguments, and handles the linkage by itself. /JU.
<code>viewzoomin_Callback</code>	Zooms in current view panel.
<code>viewzoomout_Callback</code>	Zooms out in current view panel.
<code>volume_helper</code>	Helper to find peak ejection, and empty volumes etc.
<code>volumeaxes_Buttondown</code>	Called when user has clicked volume graph, sets current timeframe to clicked point.
<code>[pos]=xyz2rlapfh(no,x,y,z)</code>	Converts from segment coordinate system to RL,AP,FH coordinate system.
<code>zoomhelper(ax,f,no,panel)</code>	Helper function to zoom image stacks. Zooming is done by changing xlim and ylim.

17 Subfunctions in `openfile.m`

This chapter describes the subfunctions of `openfile.m`

<code>newim=autocontrastpreview(im)</code>	Automatically calculates contrast settings for preview image.
<code>browsebutton_Callback</code>	Callback when user wants to browse for new folder.
<code>cancelpushbutton_Callback</code>	Dismiss the figure.
<code>numberofdirections=continueloading(files2load,numselected,directiontoload)</code>	This function is called from <code>cropbox_buttondown</code> . It continues to load the whole images stack and finally call <code>Segment</code> to tell that finished loading one image stack.
<code>cropbox</code>	Called from <code>continue loading</code> and sets up GUI to ask user to select region of interest to load.
<code>cropbox_buttondown</code>	Buttondown function for crop. This is equal to <code>select</code> .
<code>cropbox_motion</code>	Motion function for <code>cropbox</code> .
<code>exitgui</code>	Quits and exits the gui.
<code>extractvenc(dinfo)</code>	Extract <code>VENC</code> .
<code>factor_Callback</code>	Help function to split number of selected frames in to prime numbers.
<code>temp=fastdicomread(filename,startofpixeldata,numframes)</code>	

	Reads a DICOM image fast. Takes filename, and start of pixeldata.
<code>[keystruct,uniquedirections,selecteddirection]=findnumberofdirections(keystru</code>	Finds number of directions, and ask if remove unwanted directions.
<code>[usedicomcache,dicomcache]=getdicomcache(files2load,numselected)</code>	Get dicomcache.
<code>z=getfilenumber(stri)</code>	Take only the last digits of a filename if numbers exist.
<code>files2load=getfiles2load(ind2load)</code>	Gets files to load as a cell removes chache files, dicom dirs, .seg files etc.
<code>res=getinfo(s,default,varargin)</code>	Used to extract info from a structure array Can take several input arguments to check for synonyms.
<code>getpathinfo</code>	Scans directory and builds a structure of relevant information. The function populates the file list-box and stores DATA.Preview.FileList variable If there is a directory, return the number of files. If it is a file, simply the filename Later add also if it is selected.
<code>[hasdcm,description]=getseriesdescription(folder,varargin)</code>	Assumes that folder is a folder with a DICOM image series (one or more dicom files).
<code>ind=getsortingorder(keystruct)</code>	

	Gets sorting order of images given keystruct Sort the files after: * slicekey (most significant) * typekey * instancekey * triggerkey * filekey (least significant).
<code>[rotated,needstoflip,angles]=isrotated(keystruct,uniquedirections)</code>	Tests if it is a rotated image stack.
<code>keypressed(fignum,evnt)</code>	Helper function to handle keypressed events.
<code>[im,numberofdirections]=loaddicomfiles(files2load,numselected,directiontoload)</code>	This function loads the DICOM files.
<code>im=loadmatfile(files2load)</code>	Load mat file, called from continueloading.
<code>loadmultidataset</code>	Load .mat file. Sets up display and exits GUI and return back to segment.
<code>loadpreview(files2load,orientations)</code>	Loads a preview of, the function takes a filename or a pathname.
<code>im=loadpreviewdicom(file2load)</code>	Loads preview information from a DICOM file and updates DATA.Preview.
<code>im=loadpreviewmat(file2load)</code>	Extracts preview information from a .mat file sets the DATA.Preview struct.
<code>im=loadpreviewsegdicom(file2load)</code>	Extracts preview information from a .segdicom file and sets the DATA.Preview struct.
<code>im=loadpushbutton_Callback</code>	This function is called when user press load.

<code>info=metadata2info(meta)</code>	This function converts from metadata as specified by Linki ₂ ping group to info struct in Segment.
---------------------------------------	---

<code>pathlistbox_Callback(updir)</code>	This function is called when user selects something in the file/dir selection listbox. If called with on input variable then go up one directory level.
--	---

<code>plotslicelocation_Callback</code>	Plots graphical display of slicelocation of DICOM files.
---	--

<code>[im,keystruct]=preparevelocityencoded(im,keystruct)</code>	Called Loading both magnitude and phase data Creates a 5D image data to signal that is velocity encoded. Lot of extra stuff to find in what direction the flow is...
--	---

<code>[varargout]=recursedicomcache(pathname,h)</code>	Recursive version of DICOM cache operation. high level call should be without second argument.
--	--

<code>num=recursedicomcache_helper(pathname,num)</code>	Recurse down to see how many folders to do.
---	---

<code>recursedicomcachepreviewfolder_Callback</code>	Recursive version of DICOM cache operation from the folder which openfile is currently in. high level call should be without second argument.
--	---

<code>timenum=reformattime(timestr)</code>	Reformat a string with format HHMMSS.XXX.
--	---

<code>refresh_Callback</code>	Refresh GUI.
-------------------------------	--------------

<code>roisizelistbox_Callback(insize)</code>	
--	--

	Callback for ROI size selection.
<code>selectall_Callback</code>	Select all files callback.
<code>setimagetype_Callback</code>	Set image type callback.
<code>setimageviewplane_Callback</code>	Set image type callback.
<code>setimagingtechnique_Callback</code>	Set imaging technique callback.
<code>stable_Callback</code>	Callback to turn on/off stable DICOM reader.
<code>updir_Callback</code>	Fake a double click on ..
<code>verifydicomdata(keystruct,im,onlymag,rotated)</code>	Verify that correct number of slices and time-frames, slice thickness etc is correctly read from DICOM files. No output is given from this function, output is stored in the global variable DATA.Preview.
<code>stri=writeimagenormal(d)</code>	Convert numerical normal to string representation.

18 Subfunctions in `export.m`

This chapter describes the subfunctions of `export.m`

<code>exportcontour_Callback</code>	Export contour. Ask what contour to take.
<code>varargout=exportdata(doheader,includenormalized,no)</code>	This is the workhorse of export functions. - do-header tells whether to include a header. - includenormalized tells whether to include BSA normalized data. - no tells whether to export ONLY image stack no.
<code>exportimage_Callback</code>	Export current slice and timeframe to an image file.
<code>exportlv2ensight_Callback</code>	Export left ventricle to Ensight.
<code>exportmovie_Callback</code>	Function to export a movie without contours.
<code>exportmovierecorder_Callback(arg)</code>	Movie recorder GUI.
<code>exportmultiple_Callback</code>	Creaty summary of multiple matfiles in one folder.
<code>exportmultipleinfo_Callback</code>	Exports information of image stacks for a folder of mat files.
<code>ok=exportsavemovie(mov,left,right,up,down,fps)</code>	Exports a move as an avi file or a set of png-files.
<code>outdata=exportslicehelper(outdata,rowoffset,coloffset,type,x,y)</code>	Helper function to export slice based data.

`exportslicevolume_Callback`

Export slicebased volume.

`exporttthisstack(doheader)`

Export data from current image stack to clipboard.

`exporttclipboard_Callback(doheader,no)`

Export data to clipboard. Calls another function to do the export.

`exporttclipboardthisstack_Callback(doheader)`

Exports data for current image stack to clipboard.

`exportvolumecurve_Callback`

Export volume curve to clipboard.

`[outdata,ind]=header(onlyone)`

Helper function to write header when exporting data.

19 Subfunctions in roi.m

This chapter describes the subfunctions of `roi.m`

<code>removefromallbutthistimeframe(n)</code>	Remove ROI from all but this time frame.
<code>removefromthistimeframe(n)</code>	Remove ROI from this timeframe.
<code>roi_Buttondown(panel)</code>	Button down function for drawing roi.
<code>roiaddfixsize_Callback</code>	Function for adding roi with fix size.
<code>roiaddinsector_Callback(angle,width,percentfromendo,percentfromepi,numsectors,name)</code>	Add sectors in ROI, input arguments is angle of where to start, the width of the sector given as an angle, how many percent from endo the sectors will be placed, how many percent from epi the sectors will be placed, the number of sectors and finally the name under which the sector/roi will be stored.
<code>name=roiasklabel(roitename,roinamein)</code>	Lets user pick name of roi from a picklist. Input: roitename is the number of the roi to name, if this input argument is empty the choice of naming the roi to 'ROI-n' is disabled (the function can not handle a cell so if multiple rois are to be named use " as first input), roinamein is the name of the roi to be renamed (this functionality is used when renaming using template). Both input arguments are optional but if the first argument is not supplied the choice of naming the roi to 'ROI-n' is disabled.
<code>name=roiasktemplate(no)</code>	Function used for asking for name of roi template.

<code>roibar_Buttondown(type)</code>	Button down function for pressing timebar in plot flow gui. Sets button motion and button up function for changing start time and end time for calculation of flow.
<code>roibar_Buttonup(type)</code>	Button up function called after moving timebar in plot flow gui. Changes start time and end time for calculation of flow.
<code>roibar_Motion</code>	Button motion function called when moving timebar in plot flow gui. Changes start time and end time for calculation of flow.
<code>roiclearall_Callback</code>	Clear all ROIs.
<code>roicopyalltimeframes_Callback(n)</code>	Copy ROI to all time frames.
<code>roicopydownwards_Callback(n)</code>	Copy ROI one slice downward.
<code>roicopyendo_Callback</code>	Copy from endocardium to a ROI.
<code>roicopyepi_Callback</code>	Copy from epicardium to a ROI.
<code>roicopyfromotherimagestack_Callback</code>	Copy roi from other image stack. Let's user select from which image in an input dialog.
<code>roicopyscar_Callback</code>	Copy from scar to a ROI.
<code>roicopyupwards_Callback(n)</code>	Copy ROI one slice upward.
<code>roidetele_Callback(n)</code>	Delete ROI.

<code>no=roifindmag(no)</code>	Find magnitude no.
<code>roiforce_Callback</code>	Function to set force shape over time in menu checked and unchecked.
<code>roiforceapply</code>	Function to force shape over time.
<code>m=roiget(stri,arg)</code>	Gets roi either by getting closest ROI to clicked coordinates if two input arguments or by selecting in a pick list.
<code>roisign=roiguessign(nom,nop,currentroi)</code>	Guesses sign of the roi to make the net flow through the roi be positive.
<code>roihistogram_Callback</code>	Make intensity histogram from ROI with either normalized intensities or true intensities. Uses roiselector to let user decide what data to be analysed in the histogram. Also let's user decide how many bins in the histogram and if zero should be excluded.
<code>roiimportroi_Callback</code>	Import roi from image stack. Let's user select from which image in an input dialog.
<code>roilabel_Callback(x,y)</code>	Function for putting name on a roi.
<code>roiputroi_Buttondown(panel)</code>	Buttondown function for putting circular roi.
<code>[rois,timeframes,normalized]=roiselector(usealltimes,thissliceonly,template,normaliz</code>	Dialogbox in which user decides if all timeframes, only this slice and normalized intensity values should be used. All values can be changed seperately.

<code>roiSetColor_Callback(n)</code>	Change color of roi. Which roi to use is either decided by closest roi from clicked coordinate if input argument is -1 else by user selecting from a pick list.
<code>roisetLabel_Callback(n)</code>	Change name of roi. Which roi to use is either decided by closest roi from clicked coordinate if input argument is -1 else by user selecting from a pick list.
<code>roiswitchsign_Callback(m)</code>	Switch sign of roi. The sign is used when calculating flow through roi.
<code>roitemplatedelete_Callback</code>	Function to delete roi by template.
<code>roitemplatesetcolor_Callback</code>	Function to rename roi by template.
<code>roitemplatesetlabel_Callback</code>	Function to rename roi by template.
<code>roithresholdnumeric_Callback</code>	Numeric threshold inspector.
<code>roithresholdvisual_Callback</code>	Visual threshold inspector.
<code>roitoendo_Callback</code>	Copy from endocardium to a ROI.
<code>roivisualslider_Callback(arg)</code>	Gui for visual threshold.
<code>roivisualsliderkeypressed</code>	Key pressed function for visual slider.

20 Subfunctions in `utility.m`

This chapter describes the subfunctions of `utility.m`

<code>clearsegmentationmultiple_Callback</code>	Creaty summary of multiple matfiles in one folder.
<code>findindicomfiles_Callback</code>	Recursevily finds .mat files in selected directory and exports data on patient info and file location. Uses some heuristic to avoid checking all files.
<code>findinmatfiles_Callback</code>	Recursevily finds .mat files in selected directory and exports data on patient info and file location.
<code>init</code>	Calls private initialize method. Initialize utility-menu, called upon starting of Segment.
<code>[nfound,line]=numexist(outdata,rows,name,id)</code>	Helper function to findindicomfiles.
<code>utilityanonymizemultiple_Callback</code>	Anonymize multiple datasets.
<code>utilitycopyandsortfromcd_Callback</code>	Utility function to copy and sort files from CD. Uses helper function dicomsorter to do the work.

21 Subfunctions in `tools.m`

This chapter describes the subfunctions of `tools.m`

<code>addnoise_Callback</code>	Adds noise to current image stack.
<code>anonymous_Callback</code>	Makes a data set anonymous.
<code>applylight_Callback</code>	Apply current light setting to current image stack.
<code>autoesed_Callback(silent)</code>	Autodetect and store ED, and ES.
<code>copydownward_Callback(type,silent,dolv)</code>	Copy segmentation in current slice downwards and refine.
<code>copyforwardselected_Callback</code>	Copy segmentation of selected slices forward in time.
<code>copyupward_Callback(type,silent,dolv)</code>	Copies upward and refine. <code>type</code> is either <code>endo</code> or <code>epi</code> . <code>silent</code> is true if to avoid graphic update. if <code>dolv</code> false then copy <code>rv</code> .
<code>crop_Buttondown(panel)</code>	Buttondown function to crop the image stack.
<code>crop_Buttonup</code>	Buttonup function for cropping of image stacks.
<code>crop_Motion</code>	Motion function to crop the image stack.
<code>[outx,outy]=cropcontour(inx,iny,isroi)</code>	Used to crop and reinterpolate contours.

CHAPTER 21. SUBFUNCTIONS IN *TOOLS.M*

<code>disableundo(no)</code>	Disable undo function. Also copies segmentation data to make sure data is consistent.
<code>duplicateimagestack_Callback</code>	Duplicates current image stack.
<code>enableundo(no)</code>	Enables undo function in menus and icons. Also copies old segmentation data so undo is possible. This should always be called before routines that change the segmentation.
<code>enddiastole_Callback(noupdate)</code>	Change current time frame to endsystole.
<code>enddiastoleall_Callback</code>	Change current timeframe of all image stacks to diastole.
<code>endsystole_Callback(noupdate)</code>	Change current time frame to endsystole.
<code>endsystoleall_Callback</code>	Change current timeframe of all image stacks to systole.
<code>extraslice_Callback(type)</code>	Add an extra slice. Type is either 'basal' or 'apical'.
<code>flip_Callback(dim)</code>	Helper function to image stack flipping tools.
<code>flipt_Callback</code>	Helper function to flip in t direction.
<code>[a,b]=flipvars(b,a)</code>	Flip variables.
<code>flipx_Callback</code>	Flip x direction of current image stack. Need to flip both x and z to maintain a righthand system.
<code>flipx_helper</code>	Helper function to flip in x direction.

<code>flipxy_helper</code>	Interchange z and t of current image stack.
<code>flipxz_Callback</code>	Interchange z and x of current image stack.
<code>flipy_Callback</code>	Flip y directon of current image stack. Need to flip both y and z to maintain a righthand system.
<code>flipy_helper</code>	Helper function to flip in y direction.
<code>flipz_Callback</code>	Flip z direction of current image stack. Need to flip both z and x to maintain a righthand system.
<code>flipz_helper</code>	Helper function to flip in z direction.
<code>flipzt_Callback</code>	Interchange z and t for current image stack.
<code>imageinfo_Callback</code>	Displays image information of current image stack.
<code>intensitymapping_Callback</code>	Plots intensity mapping function.
<code>invertcolors_Callback</code>	Invert colors in current image stack (essentially 1-x).
<code>kspace_Callback</code>	Shows KSPACE of image (Fourier Transform).
<code>mergestacks</code>	Merge to image stacks vertically (i.e stack in z).
<code>mirrorx_Callback</code>	Mirros in x direction. Need to flip both x and z to maintain a righthand system.
<code>normalize_Callback</code>	Normalize image data of current image stack. Make sure image intensities are in the range [0..1]. Also stores in so that true image itensities can be retrieved, see <code>calctrue</code> data.

<code>precomp_Callback</code>	Function to make intensity precompensation of MR gradient echo images.
<hr/>	
<code>removeallbutedes_Callback</code>	Remove all timeframes in current image stack except diastole and systole.
<hr/>	
<code>removeallbutthistimeframe_Callback</code>	Remove all timeframes except current timeframe.
<hr/>	
<code>removecurrenttimeframe_Callback</code>	Remove current timeframe.
<hr/>	
<code>removeduplicatetimeframes_Callback</code>	Removes duplicate timeframes, asks for the number of duplicates. Two means keep every second frame.
<hr/>	
<code>removenexttimeframes_Callback</code>	Remove next timeframe and upto and including last timeframe.
<hr/>	
<code>removeprevioustimeframes_Callback</code>	Removes previous timeframe and all frames to first timeframe.
<hr/>	
<code>removesselected_Callback</code>	Remove selected image stacks from current image stack.
<hr/>	
<code>removeslices_Callback(ind,force)</code>	Remove slices from current image stack. This is the workhorse when removing slices.
<hr/>	
<code>removethis_Callback</code>	Remove current slice.
<hr/>	

<code>removetimeframes(ind)</code>	Helper function to remove timeframes. This is the workhorse when removing timeframes.
<code>removeunselected_Callback</code>	Remove unselected slices from current image stack.
<code>x=resamplehelper(f,x)</code>	Helper function to resample image stacks. factor 2 gives $x' = 2*x-0.5$ factor 3 gives $x' = 3*x-1$ factor 4 gives $x' = 4*x-1.5$ factor 5 gives $x' = 5*x-2$ factor 2.5 gives $x' = 2.5*x-1$ factor 3.5 gives $x' = 3.5*x-1.5$ factor 0.5 gives $x' = x'*0.5$ (a odd) factor 0.5 gives $x' = x'*0.5+0.5$ (a even).
<code>rotate90right_Callback</code>	Rotate current image stack 90 degrees right. Currently not working properly.
<code>setcurrenttimeframeasfirst_Callback</code>	Sets current timeframe as first time frame by cycling data in time.
<code>seted_Callback</code>	Set diastole to be current timeframe.
<code>setes_Callback</code>	Set systole to be current timeframe.
<code>setimageinfo_Callback</code>	Asks for and sets image details of current image stack.
<code>setimagetype_Callback</code>	Sets image type for current image stack.
<code>setimagingtechnique_Callback</code>	Sets imaging technique for current image stack.
<code>temporalmeanvalue_Callback(silent)</code>	Calculate temporal mean image.
<code>undosegmentation_Callback(no)</code>	Revert segmentation from undo history.

`upsampleimage_Callback` Upsamples current image stack (in slice only).

`newvol=upsampleslices(f,vol)`
Upsamples along last dimension. Limitation input must be single or double.

`upsampleslices_Callback` Upsample current image stack in slice direction.

`newvol=upsamptemporal(f,vol)`
Helper function to upsample in time. `f` is factor and `vol` is volume to upsample.

`upsamptemporal_Callback(f)`
Upsamples current image stack in time. `f` is upsampling factor.

`newvol=upsamplevolume(f,vol)`
Helper function to upsample a volume `vol`.

`viewpatientinfo_Callback(arg)`
GUI to view and adjust patient information.

`viewtrueintensity_Callback`
View true image intensities in current timeframe and slice.

22 General input output functions

This chapter describes general input and output functions used in Segment.

22.1 myadjust.m

MYADJUST objecthandle and fighandle/mygui object.
Adjust so that a message box gets to the correct screen.
fighandle is a figure handle

22.2 mybrowser.m

MYBROWSER(URL) Opens an URL platform independent.

22.3 myclientserver.m

MYCLIENTSERVER Class to create interface object to communicate with a computational

B = MYCLIENTSERVER(NAME) Sets up a computational client with the name NAME.

METHODS

DISPLAY

Overloads display.

SEND(B,FCN,CMD,ARG). Sends a non blocking message.

FCN is function handle / function that is executed upon return of message. The function handle is called with:

- 1) Name of slave
- 2) Received status, 0 = sucess
- 3) Received command
- 4) Received arguments, often empty, depends on command

Note that there is no buffering of commands, so you can not stack multiple commands. If you do not give the slave time to read the message before you send another message.

22.4 mycopyfile.m

MYCOPY Copies a file from source to dest

MYCOPY(SOURCE,DEST,[SIZ])

Should be platform independent and faster than making a system call. If really large files are used then do a system call instead. Essentially a wrapper to COPYFILE, kept for naming consistency.

See also MYDEL, MYMKDIR.

22.5 mycountunique.m

MYCOUNTUNIQUE counts the number unique elements in the input vector.

N = MYCOUNTUNIQUE(V,<tol>)

N is the number of unique elements

V is numeric input vector

tol is numeric tolerance, if omitted then set 1e-4.

See also matlab builtin unique.

22.6 mydel.m

MYDEL Delete file from disk

[OK] = MYDEL(filename)

Should be platform independent.

See also MYCOPY, RECYCLE.

22.7 mydir.m

MYDIR Extracts directory information.
. and .. are always first on the list
thumbs.cache,folders.cache,dicomdir excluded.
folders comes first

22.8 mydisp.m

Internal display function

22.9 mydispexception.m

MYDISPEXCEPTION(ME) Pretty prints a caught
exception for debugging. ME is matlab error struct
caught in a try catch clause.

See documentation on try and catch syntax.

22.10 myexecutableext.m

MYEXECUTABLEEXT Returns proper file extension for exectutables.

See also, MYREQUIREPC, MYMKDIR, MYMOVEFILE, MYCOPYFILE, MYDEL, MYDIR.

22.11 myfailed.m

MYFAILED(STRI,FIGHANDLE)
Displays an error message STRI. FIGHANDLE is
an optional alignment indicating alignment.
Difference to errordlg is the alignment and
possibility to hit return to okey the message.
Alignment string may be a fig handle or an MYGUI object.

See also MYWARNING, MYMSGBOX, MYADJUST, MYWAITBARSTART.

22.12 mygui.m

MYGUI Class for handeling GUI's in a efficeint manner

```
G = MYGUI(FIGFILENAME,<'BLOCKING'>);
```

The purpose of this class is to facilitate to generate GUI's than can also work as a temporary container data for that so that data transfer between subfunctions becomes simple. If blocking is specified then segment internal blocking figure mechanism will be used.

The function also keeps track of position for the gui to be saved later and to be able to set position of messageboxes, called with for example mywarning, mymsgbox and yesno when calling with last input argument as the mygui object.

Example:

First, reserve space for it in DATA Structure:

```
DATA.GUI.MyGUIExample = []; %in segment.m
```

In your code to initialize:

```
gui = mygui('myguiexample.fig');
```

In your subfunctions:

```
global DATA
```

```
gui = DATA.GUI.MyGuiExample;
```

Now you can use gui as an ordinary variable, and everything is stored in the GUI, and will be available to other subfunctions until the GUI is deleted. Some examples:

```
gui.myfieldname = somearray;  
gui.myfieldname1.myfieldname2.myfieldname3 = 134;  
temp = gui.myfieldname(2);  
temp = get(gui.handles.text2,'string');
```

You can also use the gui to get good alignment of the msgboxes mywarning, myfaield, mymsgbox, mymenu, mywaitbarstart and yesno.

```
myfailed('An error has occurred.',gui);
```

When closing the gui the close function can be called with
`DATA.GUI.MyGUIExample=close(DATA.GUI.MyGUIExample);`
when closing by using this syntax the position of the gui
will be saved in the struct `DATA.GUIPositions`

The position of the gui can also be saved by
`saveguiposition(gui);`
this could be used in a resize function

See also `SUBSREF`, `SUBSASGN`.

22.13 mymaximize.m

`MYMAXIMIZE(HANDLE)` Maximizes a figure or a mygui object
and aligns it to correct monitor.

22.14 mymenu.m

```
m = MYMENU(header,options,defaultstring)
m = MYMENU(header,item1,item2,item3,...)
```

Same as `menu`, but with the following additions:

- more elegant! uses keyboard to select items.
- modal display
- can use default string
- last optional argument is figure handle

22.15 mymkdir.m

```
MYMKDIR(NEWDIR)
[SUCCESS] = MYMKDIR(NEWDIR)
```

Make directory, difference to `MKDIR` is that
it does not issue an error message
if the folder already exists.

See also MYCOPY, MKDIR

22.16 mymovefile.m

[STATUS,MESSAGE,MESSAGEID] = MOVEFILE(SOURCE,DESTINATION)
moves the file in source to dest.

Same as MOVEFILE except for the last optional force argument in MOVEFILE. Essentially a wrapper for MOVEFILE, kept for naming consistency.

See also MYCOPYFILE, MOVEFILE.

22.17 mymsgbox.m

MYMSGBOX Helper function, works as MSGBOX

MYMSGBOX(STRI,TITLE,[ALIGNMENT])
STRI String to display
TITLE Title of messagebox
FIGHANDLE Optional figure handle indicating alignment

See also MYWARNING, MYMSGBOX, MYADJUST, MYWAITBARSTART.

22.18 myoptimpartial.m

Script for exhaustive search

22.19 myrequirepc.m

Z = MYREQUIREPC Returns true if PC platform.
If not PC platform, then an error message is displayed. Used to block certain functions that is not implemented for other platforms than PC.

22.20 myset.m

MYSET Same as SET, but removes NaN's in the handles.

22.21 mystubfailed.m

Displays generic error message from stubs

22.22 mywaitbarclose.m

MYWAITBARCLOSE

Closes a mywaitbar structure, and removes waitbar.

See also MYWAITBARSTART, MYWAITBARUPDATE.

22.23 mywaitbarstart.m

H = MYWAITBARSTART(NUMITERATIONS,STRING,[IGNORELIMIT],FIGHANDLE)

Similar to waitbar, but first input argument is the total number of iterations that will be performed. Second input argument is string that will be displayed, and third optional argument is a limit that if fewer iterations that this is performed, then there will be no graphical output. Another difference to standard waitbar is that no more than 20 graphical steps are displayed to minimize CPU consumption. Function returns a handle to a MYWAITBAROBJECT. FIGHANDLE is optional and indicates alignment

See also MYWAITBARUPDATE, MYWAITBARCLOSE.

22.24 mywaitbarupdate.m

H = MYWAITBARDATE(H)

Update a mywaitbarstructure, and increment counter one step. Graphic is updated depends on

setting when intializing the structure.

See also MYWAITBARSTART, MYWAITBARCLOSE.

22.25 mywarning.m

MYWARNING(STRI,HANDLE) Displays a warning message
User needs to click OK to discard message.

See also MYWARNINGNOBLOCK

22.26 mywarningnoblock.m

MYWARNING(STRI,HANDLE) Displays a warning message
User needs to click OK to discard message.

See also MYWARNINGNOBLOCK

22.27 myworkoff.m

MYWORKOFF Graphically show that calculation is
finished by restoring pointer.

22.28 myworkon.m

MYWORKON Graphically indicate that Segment is
busy by showing watch pointer.

Bibliography

- [1] E Heiberg, J Sjogren, M Ugander, M Carlsson, H Engblom, and H Arheden. Design and Validation of Segment - a Freely Available Software for Cardiovascular Image Analysis. *BMC Medical Imaging*, 10(1), 2010.
- [2] E. Heiberg. *Automated Feature Detection in Multidimensional Images*. PhD thesis, 91-85297-10-0. Linkoping universitet, Department of Biomedical Engineering, 2004.
- [3] E. Heiberg, L. Wigstrom, M. Carlsson, A. F. Bolger, and M. Karlsson. Time Resolved Three-dimensional Automated Segmentation of the Left Ventricle. In *IEEE Computers in Cardiology 2005*, volume 32, pages 599–602, Lyon, France, 2005.
- [4] E. Heiberg, H. Engblom, J. Engvall, E. Hedstrom, M. Ugander, and H. Arheden. Semi-automatic quantification of myocardial infarction from delayed contrast enhanced magnetic resonance imaging. *Scand Cardiovasc J*, 39(5):267–75, 2005.
- [5] H. Engblom, M. B. Carlsson, E. Hedstrom, E. Heiberg, M. Ugander, G. S. Wagner, and H. Arheden. The endocardial extent of reperfused first-time myocardial infarction is more predictive of pathologic Q waves than is infarct transmural: a magnetic resonance imaging study. *Clin Physiol Funct Imaging*, 27(2):101–8, 2007.